Datalähtöinen monitavoiteoptimointi teollisuudessa

Pro Gradu-tutkielma
Tapani Sipola
1976269
Matemaattisten tieteiden yksikkö
Oulun yliopisto
Syksy 2019

Sisältö

Johdanto

Optimointi tarkoittaa niiden ratkaisujen etsimistä, jotka rajoitteiden puitteissa saavuttavat halutuissa tavoitekriteereissä parhaan mahdollisen tason. Teollisuudessa sillä on hyvin suuri merkitys, sillä pienetkin suhteelliset säästöt ja parannukset ovat ison tehtaan mittakaavassa huomattavia. Tyypillisiä optimoinnin tavoitekriteerejä teollisuudessa ovat esimerkiksi pienin valmistuskustannus, resurssihukka ja paino ja suurin luotettavuus, kestävyys ja tuotto.

Kun optimointiongelmassa tavoitteita on vain yksi, optimaalinen ratkaisu on se, joka minimoi (tai maksimoi) tämän tavoitteen arvon. Yksitavoitteiset ongelmat ovat kuitenkin käytännössä harvinaisia, koska tavoitteeseen liittyy yleensä jonkinlainen kustannus tai kompromissi. Suurin osa matemaattisen optimoinnin kehityksestä on tästä huolimatta historiallisesti painottunut juuri yksitavoitteiseen optimointiin. Monitavoiteoptimoinnin tutkimus lähti käyntiin vasta 60-luvun vaihteessa, ja on kasvanut suosiossa huomattavasti 80-luvulta alkaen. [1, 3, 17]

90-luvun alussa alkoi ilmaantua useita epälineaarisia monitavoiteoptimointiongelmia, joihin senhetkiset tekniikat eivät toimineet. Näitä pyrittiin ratkaisemaan yhdistämällä monitavoiteoptimointi ja evoluutioalgoritmit. Ensimmäinen evolutiivinen monitavoiteoptimointialgoritmi oli Schafferin vuonna 1984 kehittämä VEGA, ja 90-luvulla kehitettiin useita muita. [3, 17]

Teknologian kehitys on mahdollistanut monitavoiteoptimoinnin lisäksi myös suurien datamäärien keräämisen ja tallentamisen. Tämän myötä on kehitetty uusia tapoja analysoida ja hyödyntää tätä dataa. Koneoppiminen on yksi näistä tavoista ja on viime vuosikymmeninä kasvanut keskeiseksi osaksi informaatioteknologiaa. Sen avulla on muun muassa mahdollista muodostaa tuotantoprosessin datasta malli, joka ennustaa tuotteen ominaisuuksia prosessin jälkeen.

Tämän tutkielman tarkoitus on esitellä koneopitun mallin sovittamista, ja sen hyödyntämistä monitavoitteisessa optimoinnissa, käyttäen esimerkkinä teräksen lämpökäsittelyprosessin optimointia. Tämän lisäksi vertaillaan kahta geneettistä monitavoiteoptimointialgotritmia.

1 Monitavoiteoptimointi

Kun optimointitehtävällä on monia ristiriitaisia tavoitteita, optimaalinen ratkaisu ei ole yhtä yksinkertainen määritellä kuin yksitavoitteisessa tapauksessa. Yksi ratkaisu saattaa tuottaa ominaisuuksiltaan parhaan tuotteen, mutta olla huomattavasti kalliimpi kuin muut ratkaisut, ja toinen päinvastoin. Kolmas saattaa olla kompromissi molempien tavoitteiden suhteen. Näistä ratkaisuista optimaalisen valitseminen vaatii asiantuntemusta tutkittavasta ongelmasta. Tällainen asiantuntemus on usein kokemuspohjaista, laadullista ja epäteknistä. Se, että yksi henkilö omaisi tarvitun asiantuntemuksen sekä ongelmasta että optimointimenetelmistä ei ole yleistä eikä välttämättä käytännöllistäkään. Tämän vuoksi Miettinen [12] tekee eron *analyytikon* ja *päätöksentekijän* välille. Päätöksentekijä on henkilö tai ryhmä, joka omaa ylemmän tason tietoa ongelmasta ja on usein vastuussa optimaalisen ratkaisun lopullisesta valinnasta. Analyytikko on puolestaan henkilö tai tietokoneohjelma, joka tuottaa tietoa päätöksentekijän käyttöön matemaattisin menetelmin.

Tarkastellaan monitavoitteista minimointitehtävää muodossa

$$\begin{aligned} &\text{minimoi} \quad \mathbf{f}(\mathbf{x}) = \{f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x})\} \\ &\text{rajoittein} \quad \mathbf{x} \in S \end{aligned} \tag{1}$$

missä $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ on jotakin ratkaisua kuvaava *päätösmuuttujavektori*, $f_1, f_2, \ldots, f_M : \mathbb{R}^n \to \mathbb{R}$ ovat *tavoitefunktioita* ja $S \subseteq \mathbb{R}^n$ on *sallittu alue*. Tavoitefunktion $\mathbf{f}$ lähtöjoukkoa $\mathbb{R}^n$ sanotaan päätösavaruudeksi tai hakuavaruudeksi, ja sen maalijoukkoa $\mathbb{R}^M$ tavoiteavaruudeksi. Sallittu alue sisältää kaikki harkittavissa olevat ratkaisut, ja se on tyypillisesti muotoa $S = \{\mathbf{x} \in \mathbb{R}^n \mid g_i(\mathbf{x}) \leq 0, i = 1, 2, \ldots, k\}$, missä funktiot g_i ovat *rajoitefunktioita*. Maksimointitehtävää ei tarvitse käsitellä erikseen, sillä minkä tahansa funktion $g(\mathbf{x})$ maksimikohta on funktion $f(\mathbf{x}) = -g(\mathbf{x})$ minimikohta. Jatkossa oletetaan kaikkien kohdefunktioiden arvot pyritään minimoimaan.

Debin [1] mukaan monitavoiteoptimointiin on kaksi lähestymistapaa: preferenssipohjainen lähestymistapa, ja tapa, jota hän kutsuu ideaaliseksi monitavoiteoptimointiprosessiksi. Preferenssipohjaisessa toimintamallissa analyytikko pelkistää optimointitehtävän yksitavoitteiseksi päätöksentekijältä saadun korkeamman tason tiedon avulla, ja ratkaisee pelkistetyn ongelman yksitavoiteoptimoinnin menetelmin. Yleisin ja yksinkertaisin keino tähän on painokerroinmenetelmä, jossa tavoitefunktioista muodostetaan painotettu summafunktio $f(\mathbf{x}) = \sum_{i=1}^{M} w_i f_i(\mathbf{x})$, missä reaalikertoimet w_i määrätään päätöksentekijän neuvojen mukaan. Ideaalisessa toimintamallissa järjestys on päinvastainen. Ensin analyytikko etsii joukon erilaisia potentiaalisia ratkaisuja, minkä jälkeen asiantuntija valitsee näistä pisteistä parhaiten sopivan. Ideaalisen toimintamallin etuja ovat se, että analyytikko ei tarvitse tavoitteiden

välistä preferenssitietoa ja se, että asiantuntija saa enemmän tietoa mahdollisista valinnoista. Haittana on kuitenkin usean eri ratkaisun löytämiseen liittyvä laskennallinen raskaus.

Edellä mainituilla potentiaalisilla ratkaisuilla tarkoitetaan sellaisia ratkaisuja, jotka eivät ole absoluuttisesti huonompia kuin mikään muu ratkaisu. Tätä sanotaan dominoimattomuudeksi.

Määritelmä 1.1. Piste $\mathbf{x}^*$ *dominoi* pistettä $\mathbf{x}$ (merkitään $\mathbf{x}^* \preceq \mathbf{x}$) mikäli $f_i(\mathbf{x}^*) \leq f_i(\mathbf{x})$ kaikilla $i = 1, \ldots, M$ ja $f_i(\mathbf{x}^*) < f_i(\mathbf{x})$ jollakin $i = 1, \ldots, M$.

Määritelmä 1.2. Piste $\mathbf{x} \in P$ on *dominoimaton* joukossa P, mikäli ei ole olemassa sellaista pistettä $\mathbf{x}^* \in P$, jolla $\mathbf{x}^* \preceq \mathbf{x}$. Joukko P' on joukon P *dominoimaton (osa)joukko*, jos se muodostuu joukon P kaikista dominoimattomista pisteistä. Koko sallitun alueen S dominoimatonta osajoukkoa P^* sanotaan *Pareto-optimaaliseksi* joukoksi.

Määritelmä 1.3. Jos joukolla $\underline{P}$ on olemassa ympäristö, jossa yksikään piste ei dominoi yhtään joukon $\underline{P}$ pistettä, niin joukko $\underline{P}$ on *lokaalisti Pareto-optimaalinen joukko*. Toisin sanoen, jos jollakin $\varepsilon > 0$ ei ole olemassa yhtäkään sellaista $\mathbf{y} \in \mathbb{R}^n$ ja $\mathbf{x}, \mathbf{p} \in \underline{P}$, joilla $\|\mathbf{x} - \mathbf{y}\| \leq \varepsilon$ ja $\mathbf{y} \preceq \mathbf{p}$, niin $\underline{P}$ on lokaalisti Pareto-optimaalinen joukko.

Ideaalisessa mallissa analyytikon tavoite on siis löytää Pareto-optimaalinen joukko tai sitä mahdollisimman lähellä oleva joukko. Lisäksi löydetyn joukon on katettava mahdollisimman suuri osa todellisesta Pareto-joukosta, eli Pareto-joukon jokaisen alueen lähellä on oltava ratkaisuja mahdollisimman tasaisesti. Tällöin päätöksentekijällä on käytössään mahdollisimman monta erilaista ratkaisuvaihtoehtoa. Lisäksi, koska preferenssidataa ei ole saatavilla, on löydettyjen ratkaisujen oltava mahdollisimman monimuotoisia. Dominoimatonta joukkkoa kutsutaan myös dominoimattomaksi rintamaksi, ja Pareto-joukkoa vastaavasti Pareto-rintamaksi. Usein rintama-nimityksellä viitataan nimenomaisesti kuvajoukkoon tavoiteavaruudessa.

2 Evoluutioalgoritmit

Perinteisissä haku- ja optimointialgoritmeissa, kuten esimerkiksi konjugaattigradienttimenetelmässä, kussakin iteraatiossa tutkitaan vain yhtä ratkaisua kerrallaan, edeten askel askeleelta kohti parempaa ratkaisua. Jotta löydettäisiin useita eri Pareto-optimaalisia ratkaisuja tällainen algoritmi joudutaan ajamaan monta kertaa eri parametrein. Evoluutioalgoritmit puolestaan käsittelevät ja tuottavat kerralla suuren joukon ratkaisuja, minkä takia ne soveltuvat mainiosti monitavoiteoptimointiin. Evoluutioalgritmien nimitys johtuu

siitä, että ne imitoivat eliöpopulaation kehittymisprosessia luonnossa. Myös niiden yhteydessä käytettävä termistö on lainattu biologiasta ja genetiikasta. Evoluutioalgoritmeja on useita erilaisia ja niiden tärkeimmät komponentit ovat:

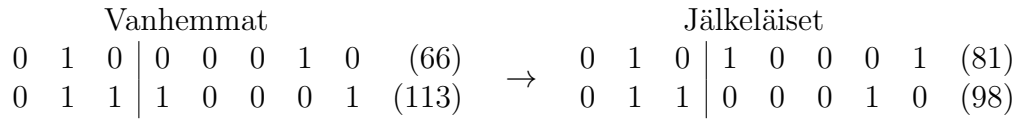
- Yksilöiden esitys,
- Kelpoisuusfunktio,
- Populaatio,
- Vanhempien valintamekanismi,
- Variaatio-operaattorit, risteytys ja mutaatio, sekä
- Seuraavan sukupolven valintamekanismi.

Yksilöllä tarkoitetaan populaatioon kuuluvaa yksittäistä ratkaisuyritelmää. Alkuperäisessä ongelmakontekstissa yksilöitä kutsutaan fenotyypeiksi, ja niiden algoritmin sisäistä esitystä sanotaan genotyypiksi. Jos esimerkiksi etsitään kokonaislukuratkaisua, sallitut kokonaisluvut olisivat fenotyyppejä. Tällöin niiden esitykseksi voitaisiin valita niiden binäärikoodaus. Fenotyyppiä 18 vastaava genotyyppi olisi tässä tapauksessa 10010. Kun puhutaan yksilöiden esityksestä, voidaan tarkoittaa joko fenotyypit genotyyppeihin liittävää kuvausta, tai genotyypin esitysmuotoa ja datarakennetta itseään. Esityksen valinta rajoittaa käytettävissä olevia funktioita ja operaattoreita, ja näin vaikuttaa evoluutioalgoritmin muiden komponenttien valintaan. Joissakin tilanteissa fenotyyppi ja genotyyppi voivat olla identtisiä.

Kelpoisuusfunktio on funktio, jolla mitataan populaation yksilöiden laatua. Optimointiongelmien yhteydessä kelpoisuusfunktio on usein sama asia kuin kohdefunktio, tai joku sen yksinkertainen muunnos. Populaatio on kaikkien algoritmin sen hetkisessä iteraatiossa tutkittavien ratkaisujen multijoukko[1]. Usein populaation määrittelyyn riittää sen koon valinta, mutta joissakin evoluutioalgoritmeissa populaatioon liittyy lisärakenteita, kuten alkioiden välisiä etäisyysmittoja ja naapurirelaatioita. Algoritmin joka iteraatiossa populaatioon tulee uusia jäseniä ja vanhoja jäseniä poistuu. Iteraatiota (ja populaatiota joka sen tuloksena saadaan) sanotaan sukupolveksi.

Vanhemmiksi kutsutaan niitä alkiota, joiden pohjalta muodostetaan uusia alkioita, eli jälkeläisiä, variaatio-operaattoreita käyttäen. Sopivalla vanhempien valintamekanismilla pyritään parantamaan jälkeläisten laatua. Vanhemmat valitaan stokastisesti siten, että mitä korkeampi kelpoisuus yksilöllä on

[1]Multijoukossa sama alkio voi esiinyä useaan kertaan



Kuva 1: Esimerkki binääriristeymästä

Vanhempi	0	1	0	0	0	0	1	0 (66)
Jälkeläinen	0	1	0	1	0	0	1	0 (82)

Kuva 2: Esimerkki binäärimutaatiosta

sitä todennäköisemmin se valitaan vanhemmaksi. Tämä voidaan suorittaa esimerkiksi turnausvalinnalla, joka on esitelty algoritmissa 2.

Variaatio-operaattoreiden tehtävä on muodostaa satunnaisia jälkeläisiä, jotka muistuttavat vanhempiaan, mutta eivät liikaa. Yhden yksilön variaatio-operaattoria sanotaan mutaatioksi ja useamman yksilön operaattoria risteytykseksi, eli rekombinaatioksi. Niihin usein liittyy parametrejä, joilla säädellään jälkeläisten hajontaa. Algoritmi voi käyttää jälkeläisten luomiseen useita eri variaatio-operaatiota.

Esimerkki 2.1. Kuvissa 2 ja 1 on esitetty yksinkertainen binäärinen risteytys ja mutaatio. Edellisessä kaksi binääristä genotyyppiä vaihtavat bittejä satunnaisesti valitun pisteen jälkeen, ja jälkimmäisessä satunnainen bitti muuttuu vastakkaiseksi. Menetelmät ovat yksinkertaisia toteuttaa, mutta niihin liittyy ongelmia. Jos genotyyppi esittää kokonais- tai reaalilukua (esim. liukulukuesityksellä), saattavat lähekkäiset luvut olla esitykseltään hyvin erilaisia. Esimerkiksi 64 ja 63 ovat vierekkäisiä kokonaislukuja, mutta niiden binääriesitykset 100000 ja 0111111 ovat täysin erilaisia. Tällöin luvusta toiseen siirtyminen vaatii että kaikki bitit vaihtuvat. Mikäli etsittävä ratkaisu on juuri tällaisessa luvussa, sen löytäminen hankaloituu. Tätä ilmiötä sanotaan Hammingin jyrkänteeksi (Hamming cliff). Yksi ratkaisu tähän ongelmaan on simuloida risteytystä ja mutaatiota stokastisesti. Mutaatiota voidaan simuloida lisäämällä fenotyyppiin esimerkiksi tasaisesti, normaalisti tai muulla tavalla jakautunut satunnaisluku. Risteytyksen simulointi onnistuu samantapaisesti kirjassa [1] esitetyllä SBX-operaattorilla[2], jossa vanhemmat joko lähenevät tai loittonevat satunnaisesti.

[2]Simulated Binary Crossover

Algoritmi 1 Evoluutiometa-algoritmi

1: Muodosta joukko ratkaisuja (aloituspopulaatio)
2: Arvioi jäsenet.
3: **while** Lopetusehtoa ei ole saavutettu **do**
4: Valitse vanhemmat.
5: Muodosta jälkeläiset valittuja vanhempia risteyttämällä ja mutatoimalla.
6: Arvioi jälkeläiset
7: Valitse seuraava sukupolvi
8: **end while**

Viimeinen komponentti evoluutiolgoritmissa on seuraavan sukupolven valintamekanismi. Uusi sukupolvi muodostetaan jälkeläisistä ja usein myös vanhemmista. Toisin kuin vanhempien valinta, uuden sukupolven valinta on tyypillisesti deterministinen. Ikäpohjaisessa valinnassa jokainen alkio pidetään populaatiossa tietty määrä sukupolvia, jonka jälkeen se poistetaan. Tämä voi äärimmäisessä tapauksessa tarkoittaa jopa sitä, että vanhemmat poistetaan joka sukupolvessa. Kelpoisuuspohjaisessa valinnassa seuraavaan sukupolveen valitaan parhaan kelpoisuuden omaavat yksilöt. Tämä takaa että paras löydetty ratkaisu säilyy aina populaatiossa, mikä nopeuttaa parhaan ratkaisun löytämistä. Parhaiden ratkaisujen säilyttämistä populaatiossa sanotaan elitismiksi. Elitismistä saattaa kuitenkin seurata ennenaikainen suppeneminen, eli algoritmin juuttuminen paikalliseen optimiin. Tämän takia kelpoisuuspohjaisen valinnan kanssa on hyvä käyttää jonkinlaista monimuotoisuuden säilytysmekanismia. Se voidaan yhdistää esimerkiksi ikäpohjaisen valinnan kanssa, tai populaatista voidaan poistaa kahdentuneet ratkaisut, tai kelpoisuutta voidaan heikentää suhteessa siihen, kuinka monta samanlaista ratkaisua populaatiossa on. [4]

Algoritmi 1 kuvaa evoluutioalgoritmien kulkua. Algoritmia suoritetaan, kunnes lopetusehto saavutetaan. Lopetusehto voi olla esim. käytetyn laskentaajan, kelpoisuusfunktion evaluaatiokertojen tai sukupolvien määrän ylärajan saavuttaminen.

2.1 Rajoitteiden käsittely

Mikäli optimointitehtävään liittyy rajoitteita sallitujen ratkaisujen suhteen, on evoluutioalgoritmin pystyttävä välttämään sallimattomia ratkaisuja. Tähän on useita erilaisia keinoja. Kun rajoitteet ovat yksinkertaisia, on kätevintä rajoittaa genotyypit sallitulle alueelle. Tämä tarkoittaa että ennen sukupolvien karsintaa rajojen ylimenevät arvot muutetaan rajoitteisiin sopi-

viksi tai poistetaan, tai että mutaatio- ja risteymäfunktiot valitsevat arvoja vain sallitulta alueelta. Jos rajoitteena toimiva funktio on monimutkainen ja raskas laskea voivat nämä keinot olla epäkäytännöllisiä. Siirtäminen sallitulle alueelle ei onnistu jos sallitun alueen rajat ovat epäselvät. Mikäli rajoitusfunktio on tiukka, ei poistaminenkaan ole käytännöllistä, sillä populaatio saattaa pienetä huomattavasti. Tällaisten rajoitteiden käsittelyyn yksinkertaisin menetelmä on asettaa rajojen ulkopuolella olevien ratkaisujen kelpoisuus alemmaksi kuin pätevistä ratkaisuista vähiten kelpoisan ratkaisun arvo. Tämän lisäksi kelpoisuutta on hyvä vähentää myös sen mukaan, kuinka paljo kukin ratkaisu on rajojen ulkopuolella.

3 Monitavoiteoptimointi evoluutioalgoritmein

Jotta evoluutioalgoritmeja voitaisiin soveltaa Pareto-rintaman hakuun, valintamekanismin on suosittava dominoimattomia ratkaisuja ja säilytettävä populaation monimuotoisuus. Tämän lisäksi valintamekanismi ei saa suosia yhtä tavoitekriteeriä muiden kustannuksella. Dominoimattomuutta silmällä pitäen yleisesti käytetty mitta ratkaisujen vertailuun on dominoimattomuustaso (non-domination rank).

Määritelmä 3.1. Joukon $Q \in S$ ensimmäinen *dominoimattomuusrintama* $\mathcal{F}^1$ on joukon Q dominoimaton rintama. Joukon Q n. dominoimattomuusrintama $\mathcal{F}^n$ on joukon $Q \setminus \bigcup_{i=1}^{n-1} \mathcal{F}^i$ dominoimaton rintama. Pisteen $\mathbf{x} \in \mathcal{F}^i$ *dominoimattomuustaso* $r(\mathbf{x}) = i$ kaikilla $i \in \mathbb{N}$.

Dominoimattomuustaso on siis sitä korkeampi, mitä useamman dominoimattomuusrintaman ”takana” ratkaisu on, eli pienempi dominoimattomuustaso on parempi. Se suosii dominoimattomia ratkaisuja eikä ole riippuvainen tavoitefunktioiden skaalauksesta, minkä vuoksi se ei myöskään suosi yhtä tavoitekriteeriä muiden kustannuksella. Dominoimattomuustaso ei kuitenkaan itsekseen osaa säilyttää populaation monimuotoisuutta. Lisäksi, koska se on diskreettiarvoinen, se ei pysty laittamaan kaikkia ratkaisuja yksiselitteiseen paremmuusjärjestykseen. Tämän takia sen kanssa käytetään aina jotain erillistä monimuotoisuuden säilyttämismenetelmää, jonka avulla myös saman dominoimattomuustason saaneet ratkaisut voidaan laittaa paremmuusjärjestykseen. Seuraavaksi esitellään kaksi monitavoiteoptimointialgoritmia, jotka kumpikin käyttävät seuraavan sukupolven valintaan dominomattomuustasoja.

3.1 NSGA-II

NSGA-II-algoritmi kehitettiin aikaisemman NSGA:n[3] pohjalta. Alkuperäistä algoritmia kritisoitiin korkean aikakompleksisuuden, elitismin puutteen ja jakoparametrin käyttämisen takia. NSGA-II kehitettiin korjaamaan nämä puutteet[2].

NSGA-II ei käytä tyypillisiä kelpoisuusfunktioita ja -arvoja. Sen sijaan NSGA-II käyttää ns. ahtausvertailuoperaattoria[4] $<_c$, jonka avulla populaation jäsenten välille luodaan paremmuusjärjestys.

Määritelmä 3.2. Olkoon $\mathbf{x} \in \mathcal{F}^i$ ja $\mathbf{y} \in \mathcal{F}^j$, missä $\mathcal{F}^i$ ja $\mathcal{F}^j$ ovat joukon Q $i.$ ja $j.$ dominoimattomuusrintamat. Piste $\mathbf{x}$ on kelpoisampi kuin $\mathbf{y}$ (merkitään $\mathbf{x} <_c \mathbf{y}$), jos $i < j$, tai jos $i = j$ ja $d_c(\mathbf{x}, \mathcal{F}^i) > d_c(\mathbf{y}, \mathcal{F}^j)$, missä d_c on ahtausetäisyysmitta (määritelmä 3.3).

Tätä vertailuoperaattoria käytetään sekä vanhempien valinnassa, että seuraavan sukupolven valinnassa. Vanhemmat valitaan kaksijäsenisen turnausvalinnan avulla, jonka k-jäseninen yleistys on esitetty algoritmissa 2.

Algoritmi 2 Turnausvalinta k jäsenellä

```
Luodaan vanhempien joukko P
while Joukossa P vähemmän kuin N alkiota do
    Valitaan populaatiosta satunnaisesti k ≥ 2 yksilöä.
    Lisätään valituista yksilöistä kelpoisin joukkoon P.
end while
```

NSGA-II:ssa kun jälkeläiset on luotu populaatiosta valituista vanhemmista, ne lisätään populaatioon. Sen sijaan, että ratkaisuille määrättäisiin jokin tietty kelpoisuusarvo tyypillisen yksiarvoisen funktion avulla, ne laitetaan paremmuusjärjestykseen ensisijaisesti dominoimattomuustason ja toissijaisesti ns. ahtausetäisyyden avulla. Tämän jälkeen populaatio kutistetaan taas alkuperäisen kokoiseksi poistamalla huonoimmat ratkaisut. Näin ollen NSGA-II on täysin elitistinen algoritmi, koska ratkaisu poistuu populaatiosta vain silloin kun populaatio on täynnä parempia ratkaisuja.

3.1.1 Nopea dominimattomuuslajittelu

NSGA-II-algoritmin ydin on nopea dominoimattomuuslajittelu. NSGA:n aikakompleksisuus oli populaation koon N suhteen $\mathcal{O}(N^3)$, mikä johtui käyte-

[3]Non-dominated Sorting Genetic Algorithm, geneettinen dominoimattomuuslajittelualgoritmi.

[4]Crowded comparison operator

Algoritmi 3 NSGA-II

1: Muodosta satunnainen aloituspopulaatio P_0, jossa N jäsentä
2: **for** $t = 1$ **to** G **do**
3: Valitse N vanhemman multijoukko P^*_{t-1} populaatiosta P_{t-1} kaksijäsenisen turnausvalinnan avulla.
4: Muodosta N jäsenen lapsipopulaatio Q_{t-1} vanhemmista P^*_{t-1} simuloidun binääristeytyksen avulla.
5: Mutatoi populaation Q_{t-1} jäseniä polynomisella mutaatiolla. [1]
6: Yhdistä populaatiot $R_{t-1} = P_{t-1} \cup Q_{t-1}$
7: Etsi dominoimattomuusrintamat $\mathcal{F}_1, \mathcal{F}_2, \ldots$ populaatiosta R_{t-1} käyttäen nopeaa dominoimattomuuslajittelua (algoritmi 4)
8: $P_t = \emptyset, i = 1$
9: **while** $|P_t| + |\mathcal{F}_i| < N$ **do**
10: $P_t = P_t \cup \mathcal{F}_i, i = i + 1$
11: **end while**
12: Valitse $N - |P_t|$ tasaisimmin levittäytynyttä ratkaisua rintamasta $\mathcal{F}_i$ populaatioon P_t käyttäen ahtausetäisyyttä $d_c(\mathbf{x}, \mathcal{F}_i)$ (määritelmä 3.3)
13: **end for**

tystä dominoimattomuuslajittelualgritmista. NSGA-II:n käyttämän nopeamman lajittelualgoritmin ansiosta sen vastaava kompleksisuus on enää vain $\mathcal{O}(N^2)$. Tämä saavutetaan pitämällä kirjaa kunkin ratkaisun $q \in Q$ dominoimista ratkaisuista S_q ja sitä dominovien ratkaisujen lukumäärästä n_q.

Algoritmi 4 Nopea dominoimattomuuslajittelu

Aseta jokaista $q \in Q$ kohti $n_q = 0$ ja $S_q = \emptyset$.
for all $q, p \in Q, p \neq q$ **do**
Jos $q \preceq p$, niin $S_q = S_q \cup p$. Muutoin, jos $p \preceq q$, niin $n_q = n_q + 1$
end for
$\mathcal{F}^1 = \{q \in Q | n_q = 0\}, k = 1$
while $\mathcal{F}^k \neq \emptyset$ **do**
$\mathcal{F}^{k+1} = \emptyset$
for all $q \in \mathcal{F}^k, p \in S_q$ **do**
$n_p = n_p - 1$
Jos $n_p = 0$, niin $\mathcal{F}^{k+1} = \mathcal{F}^{k+1} \cup p$
end for
$k = k + 1$
end while

3.1.2 Ahtausetäisyys

Toinen NSGA-II:n parannus on parametriton monimuotoisuuden säilytysmekanismi. Alkuperäinen NSGA käytti alkioiden arviointiin reaalista kelpoisuusarvoa, jonka laskemiseen tarvittiin niin sanottu jakoparametri. Ongelmana tässä lähestymistavassa oli se, että jakoparametri oli valittava jokaiselle ongelmalle erikseen, koska algoritmin suoriutuminen riippui hyvin paljon jakoparametrista ja jakoparametrin sopivuus puolestaan ongelmasta itsestään. NSGA-II käyttää monimuotoisuuden säilyttämiseen niin sanottua ahtausetäisyyttä[5].

Määritelmä 3.3. Olkoon $\mathcal{F} = \{\mathbf{x}_j\}_{j=1}^{J} \subset \mathbb{R}^n$ dominoimattomuusrintama, ja olkoon $I^m = \{I^m_{(j)}\}_{j=1}^{J}$ missä $I^m_{(j)} = f_m(\mathbf{x}_j)$ siten että $I^m_1 \leq I^m_2 \leq \cdots \leq I^m_J$ ja yhtäsuurien alkioiden järjestys joukossa I^m on sama kuin niitä vastaavien alkioiden järjestys joukossa $\mathcal{F}$. Olkoon vielä $I^m_0 = -I^m_{J+1} = -\infty$. Alkion $\mathbf{x}_j \in \mathcal{F}$ *ahtausetäisyys* on tällöin

$$d_c(\mathbf{x}_j, \mathcal{F}) = \sum_{m=1}^{M} \frac{I^m_{(j)+1} - I^m_{(j)-1}}{I^m_J - I^m_1}.$$

Toisin sanoen kullekin alkiolle $\mathbf{x}_j$ lasketaan kussakin optimoitavassa parametrissä f_m suraavaksi paremman alkion ja seuraavaksi huonomman alkion arvojen erotus $I^m_{(j)+1} - I^m_{(j)-1}$ suhteessa saman parametrin parhaan alkion ja huonoimman alkion arvojen erotukseen. Näiden suteellisten erotusten summa on ahtausetäisyys. Nopein tapa laskea ahtausetäisyys koko rintamalle $\mathcal{F}$, on laskea arvot $\frac{I^m_{(j)+1} - I^m_{(j)-1}}{I^m_J - I^m_1}$ yksi kohdefunktio kerrallaan, ja laskea nämä yhteen. Tällöin riittää järjestää alkiot suuruusjärjestykseen M kertaa, joten jos käytetään alkioiden lukumäärän suhteen aikakompleksisuudeltaan $\mathcal{O}(N \log N)$ olevaa lajittelualgoritmia, hitaimmassakin tapauksessa (eli kun koko populaatio kuuluu samaan dominoimattomuusrintamaan) ahtausetäisyyden laskeminen on aikakompleksisuusluokkaa $\mathcal{O}(MN \log N)$.

3.2 DEMO

DEMO[6] on yksitavoitteisen differentiaalievoluutioalgoritmin monitavoitteinen versio. Se käyttää ratkaisujen vertailuun samaa ahtausvertailuoperaattoria kuin NSGA-II, mutta jälkeläisten luomiseen se käyttää alkuperäisen DE-algoritmin risteytystä ja korvausta. Sen sijaan, että käytettäisiin simuloitua binääriristeymää ja polynomista mutaatiota, jokaista populaation $P \subset \mathbb{R}^n$

[5] Crowding distance

[6] Differential Evolution for Multiobjective Optimization

alkiota $\mathbf{p}_i$ kohti luodaan ns. jälkeläiskandidaatti $\mathbf{c}_i$. Tämä tapahtuu valitsemalla populaatiosta satunnaisesti alkiot $\mathbf{p}_{i_1}, \mathbf{p}_{i_2}$ ja $\mathbf{p}_{i_3}$, jotka ovat eri alkioita kuin p_i ja eri alkioita keskenään. Näistä alkioista muodostetaan mutanttivektori

$$\mathbf{v}_i = \mathbf{p}_{i_1} + F \cdot (\mathbf{p}_{i_2} - \mathbf{p}_{i_3}),$$

missä F on ns. skaalauskerroin. Tämä mutantti risteytetään vanhemman $\mathbf{p}_i$ kanssa siten, että aluksi satunnainen mutanttivektorin muuttuja korvaa vanhemman $\mathbf{p}_i$ vastaavan vastaavan muuttujan, jonka jälkeen muut muuttujat korvautuvat vastaavasti kukin risteystodennäköisyydellä p_c. Lopputuloksena saadaan kandidaatti $\mathbf{c}_i$.

Kun kandidaatti on luotu sitä verrataan vanhempansa kanssa. Jos kandidaatti ei ole vanhemman dominoima, se lisätään populaatioon, ja jos kandidaatti dominoi vanhempaa, vanhempi poistetaan. Kun kaikille vanhemmille on luotu jälkeläiskandidaatti, populaatio typistetään alkuperäiseen kokoonsa ahtausvertailuoperaattorin avulla kuten NSGA-II:ssa. [14]

Tällaisella risteytys-/valintamenetelmällä on hyviä ja huonoja puolia. Jälkeläisten ottaminen populaatioon välittömästi mahdollistaa sen että ne voivat osallistua risteytykseen saman sukupolven aikana, mikä nopeuttaa Pareto-rintaman löytymistä. Huonojen vanhempien välittömällä poistamisella on samanlainen vaikutus. Tästä kuitenkin seuraa että algoritmia ei voi rinnakkaistaa sellaisenaan. DEMO:n rinnakkasitetussa versiossa jälkeläisen luonti ja vertailu suoritetaan halutun kokoiselle ryhmälle vanhempia. Tämä heikentää välittömän populaatiomuutoksen nopeuttavaa vaikutusta, mutta koska algoritmi voidaan tällöin ajaa usealla rinnakkaisella prosessorilla, rinnakkaistus yleisesti ottaen nopeuttaa algoritmia huomattavasti.

4 Teollisen prosessin mallintaminen

Teollisen prosessin optimoimiseksi on prosessin tai sen lopputuotteen optimoitava ominaisuus pystyttävä mallintamaan matemaattisesti. Tyypillinen tapa tähän on fysikaalinen mallinnus, jossa prosessi mallinnetaan fysiikan lakien mukaan. Esimerkiksi optimoitaessa tukipilarin muotoa, materiaalifysiikan tai -tekniikan asiantuntijan on helppo laskea sen paino ja kuinka paljon se kestää rasitusta, mikäli sen materiaalin ominaisuudet tunnetaan hyvin, eivätkä käytetyt muodot ole liian monimutkaisia. Tarkka fysikaalinen mallinnus on usein kuitenkin resurssiraskasta, ja yleensä pätee vain hyvin spesifissä tilanteessa. Jos halutaan yleispätevämpi malli on mahdollista käyttää datapohjaista tilastollista mallinnusta ja koneoppimista. Nämä tavat eivät ole yleensä yhtä tarkkoja kuin fysikaalinen mallinnus, mutta niiden avulla voidaan mallintaa kokonaisia tuotantoketjuja ja useita eri tuotetyyppejä.

Algoritmi 5 DEMO

1: Muodosta satunnainen aloituspopulaatio P_0, jossa N jäsentä
2: **for** $t = 1$ **to** G **do**
3: **for** $i = 1$ **to** N **do**
4: Valitse satunnaisesti jäsenet $\mathbf{p}_{i_1}, \mathbf{p}_{i_2}$ ja $\mathbf{p}_{i_3} \in P_{t-1} \setminus \{\mathbf{p}_i\}$, joista muodostetaan mutanttivektori $\mathbf{v}_i = \mathbf{p}_{i_1} + F \cdot (\mathbf{p}_{i_2} - \mathbf{p}_{i_3})$
5: Muodosta jälkeläiskandidaatti c jäsenelle $p_i \in P_{t-1}$
6: Jos $p_i \preceq c$ niin c poistetaan. Jos $c \preceq p_i$ niin p_i korvataan jälkeläisellä c. Muussa tapauksessa c lisätään populaatioon P_{t-1}
7: **end for**
8: Etsi dominoimattomuusrintamat $\mathcal{F}_1, \mathcal{F}_2, \ldots$ populaatiosta P_t
9: $P_{t+1} = \emptyset, i = 1$
10: **while** $|P_{t+1}| + |\mathcal{F}_i| < N$ **do**
11: $P_{t+1} = P_{t+1} \cup \mathcal{F}_i, i = i + 1$
12: **end while**
13: Valitse $N - |P_{t+1}|$ tasaisimmin levittäytynyttä ratkaisua rintamasta $\mathcal{F}_i$ populaatioon P_{t-1} käyttäen ahtausetäisyyttä (määritelmä 3.3)
14: **end for**

Fysikaaliset mallit usein pätevät vain yhteen tuotteeseen yhdessä prosessivaiheessa.

Jotta koneoppimis- ja muut tilastolliset menetelmät toimisivat hyvin, on prosessin säätöparametreistä ja tuotteen mallinnettavista ominaisuuksista on oltava riittävästi dataa. Useimmilla teollisuusyrityksillä on kuitenkin järjestelmiä, jotka keräävät ja säilyttävät dataa tuotantoprosesseista, joten datan riittävyys ei yleensä ole ongelma. Päinvastoin yrityksillä on usein vaikeuksia saada suurta tietomääräänsä hyötykäyttöön. Tähän ongelmaan tiedonlouhintamenetelmät, kuten koneoppiminen, ovat luonnollinen ratkaisu, ja teollisuus on hyötynyt näistä useilla eri osa-alueilla, kuten aikataulutuksessa, virheentunnistuksessa, ennakoivassa ylläpidossa, suunnittelussa, tuotannossa ja laadunvalvonnassa. [8]

4.1 Koneoppiminen

Arthur Samuel määritteli koneoppimisen tieteenalaksi, joka antaa tietokoneille kyvyn oppia ilman erillistä ohjelmointia [13]. Käytännössä tämä tarkoittaa sitä että koneen on löydettävä annetusta datasta mahdollisimman paljon hyödyllistä tietoa mahdollisimman vähällä ohjauksella. Sen tunnetuimpia käytännön sovelluksia ovat roskapostisuodattimet, kääntäjät, hakukoneet ja konenäkö.

Koneoppiminen voidaan jakaa tietokoneen saaman palautteen perusteella karkeasti kolmeen tyyppiin: ohjattuun ja ohjaamattomaan oppimiseen sekä vahvistusoppimiseen. Ohjaamattomassa oppimisessa tietokoneelle ei anneta palautetta, vaan ainoastaan data, josta koneen on itsenäisesti löydettävä rakenteita. Esimerkiksi klusteroinnissa tietokoneen on itsenäisesti jaoteltava data eri luokkiin siten, että "samanlaiset" alkiot ovat samassa luokassa. Näin voidaan saada käsitys datassa mahdollisesti esiintyvistä rakenteista. Erityisesti suurten datajoukkojen tutkimisessa klusterointi on erittäin suosittu menetelmä. Vahvistusoppimisessa tietokone toimii vuorovaikutuksessa jonkin ympäristön kanssa. Ympäristö palkitsee konetta hyvistä toiminnoista ja/tai rankaisee huonoista toiminnoista. Tietokone pyrkii maksimoimaan saadun palkinnon määrän. Tällaista oppimista voidaan käyttää esimerkiksi puheentunnistusohjelmissa, joissa käyttäjä kertoo koneelle milloin se on tulkinnut puhekomennon väärin, ja opetettaessa tietokonetta pelaamaan peliä, milloin palkintona voi toimia esimerkiksi pistemäärä, tai se voittiko vai hävisikö tietokone. Vahvistusoppimista voidaan käyttää myös teollisuudessa ennustemallien opettamiseen, mutta mallin opettaminen tuotannossa ei ole suositeltavaa, jos sen on tarkoitus ohjata prosessia, joka on kriittinen tuotannon kannalta. [15]

Ennustemallien kannalta kiinnostavin on kuitenkin ohjattu oppiminen. Siinä datan lisäksi tietokoneelle annetaan jokaista datapistettä $\mathbf{x}_i$ kohti tuntemattoman funktion arvo $f(\mathbf{x}_i)$. Näistä tietokoneen on muodostettava funktiota f estimoiva funktio $\hat{f}$. On kuitenkin vaikea tietää onko $\hat{f}$ hyvä estimaati muualla kuin pisteissä $\mathbf{x}_i$, koska funktion f arvoja ei tunneta näiden arvojen ulkopuolella. Kuitenkin mitä enemmän ennustettavan pisteen ympäristössä on havaintopisteitä, sitä lähemmäs todellista arvoa päästään. Tämän vuoksi on tärkeää, että opettamiseen käytetty data kattaa mahdollisimman laajan alueen, koska tällöin malli on hyvin yleistettävissä. Jos yritetään tehdä ennusteita pisteille jotka ovat liian kaukana opetusdatan pisteistä ennustustulokset ovat useimmiten hyvin heikkoja.

4.1.1 GBM

Ohjatussa oppimisessa keskeinen ongelma on ns. piirteiden valinta (feature selection), jossa opetusdatan muuttujista poistetaan ne muuttujat (eli piirteet), jotka eivät sisällä mallin kannalta oleellista tietoa tai sisältävät samaa tietoa kuin joku toinen muuttuja. Tämä yksinkertaistaa mallia, keskittää sen sovituksen oleelliseen informaatioon ja häiriötekijöiden vähetessä mallin yleistettävyys paranee.Käsiteltävässä lämpökäsittelydatassa on suuri määrä muuttujia, joista moni vaikuttaa ennustettaviin arvoihin.

GBM[7] on koneoppimisen menetelmä, jolla pyritään ennustamaan halutun vasteen käyttäytymistä siihen liittyvien syötteiden perusteella. Vasteesta muodostetaan malli algoritmille annetun syöte-vastepareista koostuvan opetusaineiston perusteella. Tätä sanotaan ohjatuksi oppimiseksi. GBM-menetelmässä, kuten muissa boosting-menetelmissä, malli muodostetaan useista yksinkertaisista parametrisista funktioista $h(\mathbf{x}, \mathbf{a})$, missä $\mathbf{x}$ on syöte ja $\mathbf{a}$ on parametrivektori. Näitä funktioita kutsutaan heikoiksi oppijoiksi. Heikoksi oppijaksi kelpaa mikä tahansa parametrinen funktio, jolla pystytään ennustamaan dataa paremmin kuin satunnaisella arvauksella. Hyödyllinen, mutta ei pakollinen ominaisuus on myös laskennallinen keveys. Heikot oppijat sovitetaan yksi kerrallaan siten, että jokainen uusi malli keskittyy siihen dataan, jota aikaisemmat mallit ennustavat huonosti.

Olkoon $\{y_i, \mathbf{x}_i\}_1^N$ opetusaineisto, jossa $\mathbf{x}_i$ on syötevektori ja y_i sen vaste. Mallin lähtökohdaksi valitaan vakiofunktio F_0, joka minimoi summan $\sum_{i=1}^N L(y_i, F_0(\mathbf{x}_i))$, missä L on jokin tappiofunktio. Tyypillisiä valintoja tappiofunktioksi ovat neliövirhe $(y-F)^2$ ja absoluuttinen virhe $|y-F|$, kun $y \in \mathbb{R}$ ja negatiivinen binomi log-uskottavuus $\log(1 + e^{-2yF})$, kun $y \in \{-1, 1\}$. Neliövirheen tapauksessa $F_0(\mathbf{x})$ on vasteiden y_i aritmeettinen keskiarvo. Tämän jälkeen lasketaan joka iteraatiossa $m = 1, \ldots, M$ niin sanotut pseudoresiduaalit

$$\tilde{y}_i = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}, i = 1, \ldots, N. \tag{2}$$

Tämän jälkeen sovitetaan uusi heikko oppija minimoimalla

$$\mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^{N} [\tilde{y}_i - \beta h(\mathbf{x}_i; \mathbf{a})]^2 \tag{3}$$

$$\rho_m = \arg\min_{\rho} \sum_{i=1}^{N} L(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i; \mathbf{a}_m)) \tag{4}$$

ja lisätään se malliin

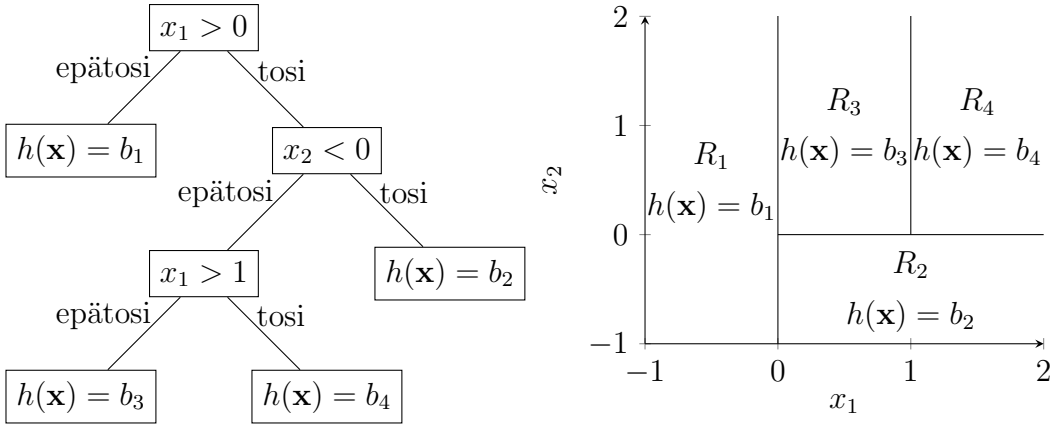
$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}_i) + \rho_m h(\mathbf{x}; \mathbf{a}_m). \tag{5}$$

GBM:n ominaisuudet riippuvat paljon käytetystä heikosta oppijasta. Yleensä tähän tarkoitukseen käytetään ns. päätöspuita. Nimitys johtuu funktion graafisen esityksen ulkonäöstä (kuva 3 vasemmalla). Puuta lähdetään tutkimaan ylimmästä solmusta eli ns. juurisolmusta, ja edetään puuta alaspäin

[7]Gradient Boosting Machine. Joskus käytetään myös nimitystä Generalized Boosted Model.

Algoritmi 6 Gradient Boosting - algoritmi [5]

1: $F_0(\mathbf{x}) = \arg\min_\rho \sum_{i=1}^{N} L(y_i, \rho)$
2: **for** $m = 1$ **to** M **do**
3: $\quad \tilde{y}_i = -[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}, i = 1, \ldots, N$
4: $\quad \mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^{N} [\tilde{y}_i - \beta h(\mathbf{x}_i; \mathbf{a})]^2$
5: $\quad \rho_m = \arg\min_\rho \sum_{i=1}^{N} L(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i; \mathbf{a}_m))$
6: $\quad F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}_i) + \rho_m h(\mathbf{x}; \mathbf{a}_m)$
7: **end for**



Kuva 3: Esimekki nelilehtisen päätöspuun graafisesta esityksestä ja sitä vastaavasta lähtöavaruuden osioinnista.

joko vasempaan tai oikeaan haaraan riippuen siitä täyttääkö funktion argumentti $\mathbf{x}$ haaran ehdon vai ei. Tätä toistetaan jokaisessa solmussa kunnes saavutaan solmuun jossa ei ole haaroja eli ns. lehteen. Päätöspuufunktion arvon alkiolle $\mathbf{x}$ määrää sen lehden arvo johon päädyttiin. Päätöspuun monimutkaisuutta on helppo säätää rajoittamalla haarasolmujen määrää.

Matemaattisesti esitettynä J-lehtinen päätöspuu on additiivinen funktio

$$h(\mathbf{x}; \{b_j, R_j\}_{j=1}^{J}) = \sum_{j=1}^{J} b_j \, \mathbb{1}_{R_j}(\mathbf{x}),$$

jossa $\mathbf{x} \in \mathbb{R}^n$ on syöte ja $\{b_j, R_j\}_{j=1}^{J}$ ovat määrittelyparametrit. Joukot $R_j \subset \mathbb{R}^n$ ovat päätöspuukuvion määrittämiä erillisiä joukkoja, jotka peittävät lähtöjoukon $\mathbb{R}^n$. Funktiot $\mathbb{1}_{R_j} : \mathbb{R}^n \to \{0, 1\}$ ovat niiden indikaattorifunktiot, eli ne saavat arvon 1 kun $\mathbf{x} \in R_j$ ja muutoin arvon 0.

Päätöspuiden sovittaminen pseudoresiduaaleihin (algoritmin 6 rivi 4) tapahtuu askeleittain. Aluksi puun sovittamista varten valitaan opetusaineis-

tosta satunnainen osajoukko, jonka koko on tyypillisesti 40-80 % koko opetusaineiston koosta[8]. Tämä parantaa mallin yleistettävyyttä ja nopeuttaa laskentaa.[7] Merkitään valittujen alkioiden indeksejä joukolla $T \subset \{1, \ldots, N\}$. Sovittamiseen käytetään neliöpoikkeamaa

$$S(A) = \sum_{i \in A} (\tilde{y}_i - \bar{y})^2,$$

missä $A \subseteq T$ ja $\bar{y}$ on keskiarvo

$$\bar{y} = \frac{1}{|A|} \sum_{i \in A} \tilde{y}_i.$$

Kutakin muuttujaa $x_i, i = 1, \ldots, n$ kohti etsitään jakopiste t_i, joka minimoi summan

$$S_i(t_i) = S(\{t \in T | x_{i,t} \leq t_i\}) + S(\{t \in T | x_{i,t} > t_i\}),$$

missä $x_{i,t}$ on alkion t arvo muuttujalle x_i. Tämän löytämiseksi riittää tarkastella suuruusjärjestyksessä peräkkäisten x_i arvojen puolivälejä. Kun jokaiselle muuttujalle on tiedossa optimaalinen jako, näistä pienimmän summan omaava jako valitaan puun ensimmäiseksi solmuksi. Lisäksi merkitään muistiin kuinka paljon solmu vähensi keskineliöpoikkeamaa, eli erotus $S(T) - S_i(t_i)$. Tämä arvo on hyödyllinen myöhemmin kun etsitään mallin vaikuttavimpia tekijöitä.

Syntyneille aliryhmille $T_{[x_i \leq t_i]}$ ja $T_{[x_i > t_i]}$ suoritetaan sama jako-optimointi jokaisen muuttujan suhteen, ja kaikkien aliryhmien kokonaisneliöpoikkeamaa eniten vähentävä jako valitaan seuraavaksi solmuksi merkiten muistiin paljonko keskineliöpoikkeama väheni edellisestä jaosta. Tätä toistetaan kunnes on muodostettu haluttu määrä solmuja.

Päätöspuiden käytöllä on useita etuja tiedonlouhinnan kannalta. Yksi näistä on ns. sisäinen piirrevalinta. Jokaiselle muuttujalle x_i voidaan laskea, kuinka paljon sen suhteen tehdyt jaot ovat vähentäneet mallin keskineliöpoikkeamaa. Tämän tiedon avulla nähdään helposti mitkä muuttujat vaikuttavat vasteeseen paljon ja mitkä eivät merkittävästi. Tällöin mallista on helppo karsia pois merkityksettömät muuttujat.

Toinen päätöspuiden etu on että malli pystyy käsittelemään puuttuvaa dataa. Kun puu jaetaan muuttujan perusteella, laitetaan kyseisen muuttujan puuttuva data omaksi haarakseen, jolle lasketaan oma keskiarvo. Tällöin pystytään hyödyntämään myös alkioita, joilla ei ole dataa kaikilta muutujilta. Kun teollisen prosessin data sisältää tietoa useasta erilaisesta tuotteesta,

[8]Tästä käytetään nimitystä bagging tai bootstrap aggregating.

saattaa olla että joiltakin tuotteilta puuttuvat tietyt mittaukset ja asetukset kokonaan. Päätöspuut mahdollistavat sen, että tällaisia alkioita ei välttämättä tarvitse poistaa tai käsitellä erikseen. Tämä saattaa kuitenkin usein olla järkevää, varsinkin jos tuotteet ovat hyvin erilaiset.

4.1.2 Yleistetty additiivinen malli

Yleistetyssä additiivisessa mallissa, eli GAM[9]-mallissa, yhdistyy additiivisen mallin joustavuus ennustettavan riippuvuuden muodon suhteen ja yleistetyn lineaarisen mallin soveltuvuus useille erilaisille vasteen jakaumille. Olkoon Y satunnaismuuttuja, jonka tiheysfunktio kuuluu eksponentiaaliseen jakaumaperheeseen. GAM on tällöin muotoa $g(\mu) = \eta$, missä

$$\eta = s_0 + \sum_{j=1}^{p} s_j(X_j),$$

on mallin systemaattinen osa, $\mu = E(Y|X_1, X_2, \ldots, X_p)$ ja g on niin sanottu linkkifunktio. Systemaattisessa osassa $s_0 = E(Y)$ ja $s_1, s_2, \ldots, s_p$ ovat sileitä funktioita joilla $E(s_i(X_i)) = 0$. Yleensä linkkifunktio valitaan vasteen todennäköisyysjakauman mukaan. Tyypillisimpiä valintoja ovat identiteettifunktio normaalijakautuneelle muuttujalle ja logit-funktio $g(\mu) = ln(\frac{\mu}{1-\mu})$ Bernoullijakautuneelle kaksiluokkaiselle muuttujalle. Hajontamallissa vasteelle käytetään Gamma-jakaumaa, koska χ^2-jakauma, jonka mukaan neliövirhe on jakautunut, on Gamma-jakauman erikoistapaus.

GAM voidaan sovittaa usealla eri menetelmällä. Hastie ja Tibshirani [9] käyttivät alunperin niin sanottua takaisinsovitusalgoritmia (backfitting algorithm). Siinä kukin funktio estimoidaan aluksi nollaksi, jonka jälkeen joka kierroksella yksi funkitoista f_i sovitetaan ennustamaan osittaisresiduaaleja $R_i = Y - s_0 - \sum_{j \neq i} s_j(X_j)$, eli sitä osaa vasteen vaihtelusta, jota muut funktiot eivät selitä. Takaisinsovituksen etuina ovat laskennallinen keveys ja se, että funktiot s_i voidaan sovittaa lähes millä tahansa menetelmällä. Hastie ja Tibshirani käyttivät tähän siloitusmenetelmiä kuten esimerkiksi liukuvaa lineaarista sovitetta ja ydinestimointia. [9, 18]

Käytetystä sovitusmenetelmästä riippumatta on järkevää rajoittaa funktioiden s_i estimointia jollakin tavalla laskennan yksinkertaistamiseksi. Yksi tapa on valita joukko kantafunktioita, joista funktiot s_i muodostetaan lineaarikombinaatioina.

Tämän tutkielman yhteydessä on hajontamallin sovittamiseen käytetty R-ohjelmiston mgcv-pakettia, joka tukee useita erilaisia kantafunktioita.

[9]Generalized Additive Model

Näistä käytettäväksi on valittu sakotettu kuutiospline[10]. Splinefunktio on paloittain määritelty funktio joka on jatkuva toiseen derivaattaansa saakka. Palojen yhtymäkohtia sanotaan solmukohdiksi, ja ne yleensä valitaan tasavälein. Funktion sakottamisella tarkoitetaan sitä että minimoitavaan neliövirhesummaan lisätään sakkotermi

$$\lambda \int [s_j''(x)]^2 dx,$$

missä λ on niin sanottu siloitusparametri. Sopiva λ etsitään käyttämällä validointimenetelmiä kuten esimerkiksi ristiinvalidointia. Sakottaminen ja silotusparametrin ristiinvalidointi vähentävät mallin ylioppimista. [18]

4.2 Hylkäystodennäköisyyden ennustaminen

Teollisessa prosessissa tuotteen tärkeimmille mitattaville ominaisuuksille on usein määritelty minimi ja maksimiarvot tai toleranssit. Mikäli lopputuotteen ominaisuudet eivät ole näiden rajojen sisällä, tuote joudutaan käsittelemään uudelleen, myymään halvemmalla tai vähemmän vaativalle asiakkaalle tai hylkäämään kokonaan. Oli valinta mikä tahansa, laatupoikkeaman seurauksena tuotteen tuottama voitto pienenee. Hylkäyksen riskiä voidaan pienentää tekemällä muutoksia tuotantoprosessiin, mutta usein tämä myös lisää kustannuksia. Tällöin on hyödyllistä pystyä ennustamaan tuotteen hylkäystodennäköisyys. Tässä tutkielmassa hylkäystodennäköisyyden ennustamiseen käytetään artikkelissa [10] esiteltyä menetelmää, jonka kokonaisprosessi ja käytetyt mallinnusmenetelmät esitellään tässä kappaleessa pääpiirteittäin.

Oletetaan että mallin vaste on reaalinen satunnaismuuttuja $\mathcal{Y}$, jonka realisaatiot ovat y_i. Jos vasteita on useampi, kuten teräksen mekaanisten ominaisuuksien tapauksessa, kukin niistä mallinnetaan erikseen. Olkoot $\mathbf{x}_i$ realisaatioita y_i vastaavat selittäjävektorit. Selittäjävektorin alkiot voivat olla reaalisia, kokonaislukuja tai binäärisiä. Oletetaan, että vasteen $\mathcal{Y}$ ehdollinen jakauma on vektorin $\mathbf{x}_i$ funktio, ja

$$y_i = \mu_i + \sigma_i \varepsilon_i, \varepsilon_i \sim F_i,$$

missä $\mu_i = \mu(\mathbf{x}_i), \sigma_i = \sigma(\mathbf{x}_i)$ ja $F_i = D(\mathbf{x}_i)$, missä D on virhetermin jakaumafunktio. Usein voidaan mallin yksinkertaistamiseksi olettaa että $F_i = F$ kaikilla i. Tarkoituksena on ennustaa tarkasti hylkäystodennäköisyys $P_i = 1 - P(Y_i^{min} \leq y_i \leq Y_i^{max})$, missä Y_i^{min} ja Y_i^{max} ovat vasteen sallitun alueen rajat.

[10]Penalized cubic spline

Aluksi sovitetaan odotusarvon ennustefunktio $\hat{\mu}(\mathbf{x})$, eli funktion $\mu(\mathbf{x})$ estimaattori. Tähän voidaan käyttää useita eri tilastotieteen ja koneoppimisen menetelmiä. Tämän tutkielman tutkimusongelman kanssa on käytetty GBM-mallia, joka esitellään kappaleessa 4.1.1.

Keskihajonnan fuktiota estimoiva σ_i voidaan myös valita usealla tavalla. Pelkän odotusarvon ennusteen $\hat{\mu}$ ja vakiohajonnan $\sigma_i = \sigma \forall i$ avulla pystytään muodostamaan suhteellisen hyvä ennuste hylkäystodennäköisyydelle, mutta artikkelissa [10] on osoitettu, että hajonnan mallintaminen tilastollisesti usein parantaa hylkäysennusteen tarkkuutta vakiohajontamalliin nähden. Tämä johtuu siitä, että teollisuussovelluksissa tuotekohtaisessa ennustusvirheessä saaattaa olla merkittäviä eroja eri tuoteryhmien välillä, ja nämä erot riippuvat valmistusparametreista. Hajonnan mallintamisen tarkoitus on löytää tällaisia systemaatisia vaihteluja hajonnan suuruudessa muuttujien tai ennusteen suhteen.

Odotusarvon ennustefunktiosta lasketaan residuaalit $\hat{\varepsilon}_i = y_i - \hat{\mu}(\mathbf{x}_i)$, joista muodostetaan hajonnan ennustamiseen käytetty vaste, tyypillisesti $\hat{\varepsilon}_i^2$. Odotusarvon ja hajonnan mallintamisessa on ongelmana se että odotusarvomallin ennuste on vääristynyt havaintoja kohti, minkä vuoksi neliöresiduaalien $\hat{\varepsilon}_i^2$odotusarvo on aina todellista hajontaa σ_i^2 pienempi. Artikkelissa [10] ehdotetaan tämän harhan korjaamiseksi skaalattua neliöresiduaalia $\hat{\varepsilon}_i^2(1 + \Delta)$, missä Δ on opetusdatan ja testidatan keskineliövirheiden suhde, eli

$$\Delta = \left[\frac{1}{N_E} \sum_{i \in \mathbf{E}} \hat{\varepsilon}_i^2 \right] / \left[\frac{1}{N_V} \sum_{i \in \mathbf{V}} \hat{\varepsilon}_i^2 \right].$$

[10]

Koska odotusarvomallin residuaaleissa satunnaisvirheen osuus on yleensä huomattavasti suurempi kuin systemaattisen virheen osuus, ei ole realistista odottaa että hajontaa voitaisiin ennustaa tarkasti. Tämän takia on hyvä käyttää mahdollisimman yksinkertaista mallia, joka kuitenkin pystyy käsittelemään epälineaarisia yhteyksiä. Näistä syistä hajonnan mallinnukseen on tässä tutkielmassa käytetty GAM-mallia.

5 Tutkimusongelma

5.1 Teräksen valmistus

Teräksen tärkeimpiä laatukriteerejä ovat lujuus ja iskusitkeys. Lujuudella tarkoitetaan teräksen kykyä vastustaa vääntymistä ja venymistä. Tyypillinen tapa mitata lujuutta on vetokoe, jossa koesauvaa vedetään kahtaalle kasvavalla voimalla kunnes se venyy ja murtuu. Voimaa, jolla sauvassa alkaa tapahtua pysyvää venymistä sanotaan myötölujuuudeksi, ja voimaa, jolla sauva murtuu sanotaan murtolujuudeksi. Tämä suhteutetaan koesauvan poikkipinta-alaan ja sen mittayksikkö on näin ollen N/mm^2. Iskusitkeydellä tarkoitetaan teräksen kykyä vastustaa äkillisten iskujen aiheuttamia murtumia. Sitä mitataan standardoidulla Charpy-V-kokeella, jossa lovettu koesauva murretaan määräkorkeudelta pudotetulla heilurilla. Sauvan absorboima energia päätellään heilurin pudotuskorkeuden erosta sen iskun jälkeiseen nousukorkeuteen. Kun testi suoritetaan matalassa lämpötilassa, esimerkiksi -40 astetta, iskusitekys on huomattavasti pienempi kuin teräksen ollessa huoneenlämpötilassa, joten eri lämpötilossa tehtyjä testejä ei voi välttämättä suoraan verrata keskenään. [11]

Sitkeys ja lujuus ovat osittain ristiriitaisia suureita. Joustava teräs kestää kovempia iskuja, mutta venyy helpommin, ja luja teräs päinvastoin on hauraampaa, mutta ei veny. Käyttötarkoituksesta riippuen terästuote vaatii erilaisen sitkeys-lujuusyhdistelmän, ja tuotteen tilannut asiakas asettaa yleensä kummallekin haluamansa raja-arvot. Teräksen mekaaniset ominaisuudet pyritään muuttamaan halutun laisiksi erilaisilla lämpökäsittelyillä.

5.1.1 Normalisointi ja nuorrutus

Normalisointi on teräksen lämpökäsittely jonka tarkoitus on lisätä teräksen sitkeyttä ja tasata sen mekaaniset ominaisuudet. Normalisoinnissa teräs kuumennetaan jopa 900 asteeseen, ja annetaan sen jälkeen jäähtyä vapaasti. Teräksen kiderakenne muuttuu tällöin hienorakeisemmaksi, minkä ansiosta siitä tulee sitkeämpää. [16]

Karkaisu on toinen teräksen lämpökäsittely, jossa teräskappale kuumennetaan yli 800 asteiseksi ja jäähdytetään nopeasti, eli sammutetaan, käyttämällä esimerkiksi vettä tai öljyä. Tämä lisää teräksen lujuutta huomattavasti, mutta tekee siitä haurasta. Karkaisun jälkeen suoritetaan päästö, jossa teräs kuumennetaan 200 asteiseksi tai kuumemmaksi ja annetaan jäähtyä. Tämä laskee teräksen lujuutta, mutta tekee siitä hyvin sitkeää. Kun päästö suoritetaan korkeammassa lämpötilassa, 500-700 asteessa, tätä sanotaan nuorrutukseksi. Yhdessä näitä kahta vaihetta kutsutaan karkaisu-päästöprosessiksi. Mi-

tä korkeammassa lämpötilassa päästö suoritetaan ja mitä pidempään terästä pidetään tässä lämpötilassa sitä sitkeämpää (ja pehmeämpää) teräksestä tulee. Mikäli pitoaika ja lämpötila eivät ole riittäviä, teräs saattaa jäädä liian hauraaksi ja lujaksi. Toisaalta mitä kuumemmassa ja mitä pidempään tuotetta nuorrutetaan, sitä kalliimmaksi prosessi tulee ja pahimmassa tapauksessa teräksestä saattaa tulla liian pehmeää. [16]

5.2 Hylkäystodennäköisyys

Jos terästuotteen mekaaniset ominaisuudet eivät ole asiakkaan vaatimissa rajoissa, se joudutaan käsittelemään uudelleen tai jopa romuttamaan. Myös tämä aiheuttaa paljon ylimääräisiä kuluja. Tämä tarkoittaa, että on mahdollista saada aikaan huomattavia säästöjä, mikäli löydetään oikea tasapaino hylkäysriskin ja tuotantokustannusten välillä. Jotta tähän kyettäisiin, on ensin pystyttävä arvioimaan hylkäysriskiä mahdollisimman tarkasti. Hylkäysriskiä on tämän tutkielman yhteydessä pyritty ennustamaan ohjatun oppimisen menetelmin, ja tätä käsitellään tarkemmin kappaleessa 4.2.

Se että hylkäysriski pystytään ennustamaan tarkasti ei kuitenkaan riitä optimaalisen ratkaisun löytämiseen. Tarvitaan tieto myös parametrien säädön kustannuksista ja säästöistä ja hylkäyksen aiheuttamista kustannuksista. Näitä ei kuitenkaan tämän ongelman yhteydessä ollut saatavilla. Tämän takia kustannukset minimoivia parametrejä ei voida löytää suoraan, vaan on turvauduttava monitavoiteoptimointiin.

5.3 Optimointiongelma

Ensimmäisenä on mietittävä optimointiongelman muotoilua. Yksinkertaisin muotoilutapa olisi kolmitavoitteinen ongelma, jossa minimoidaan pitoaika, pitolämpötila ja hylkäysriski. Tämä lähestymistapa antaa suurimman mahdollisen määrän tietoa vaihtoehtoja vertailevalle asiantuntijalle. Hylkäysriskin minimoiminen tuo kuitenkin mukanaan ongelmia. Ensinnäkin kolmannen parametrin lisääminen tekee kohdeavaruudesta kolmiulotteisen ja Pareto-rintamasta kaksiulotteisen. Tällöin koko rintaman kuvaamiseen tarvitaan suurempi populaatio, mikä kasvattaa algoritmin suoritusaikaa, algoritmin etsintäteho jakautuu isommalle alueelle ja algoritmin suppeneminen Pareto-rintamaa kohti hidastuu. Tämän lisäksi, koska monintavoiteoptimointialgoritmit pyrkivät mahdollisimman suureen monimuotoisuuteen, löydetty rintama saattaa sisältää hyvin paljon ratkaisuja, jotka hyväksyvät aivan liian suuren riskin hyvin pienen säästön takia, tai päinvastoin käyttävät paljon resursseja kutistaakseen jo tarpeeksi pientä hylkäystodennäköisyyttä enstisestään. Molemmissa tapauksissa ratkaisuja keskittyy alueille mistä kustannus-

ten suhteen optimaalisen ratkaisun löytyminen on hyvin epätodennäköistä, mikä myös syö etsintätehoa. Kolmas ongelma on löydetyn Pareto-rintaman esittäminen. Kolmiulotteinen kuva on vaikeampi esittää ja tulkita oikein kuin kaksiulotteinen.

Näistä syistä on järkevämpää minimoida vain pitoaika ja -lämpötila, ja käyttää hylkäystodennäköisyyttä rajoitteena. Matemaattisesti ongelma muotoillaan siis

$$\begin{aligned} &\text{minimoi} \quad f_1(\mathbf{x}) = x_1, f_2(\mathbf{x}) = x_2 \\ &\text{rajoittein} \quad \mathbf{x} \in S = \{\mathbf{x} \in [a_{min}, a_{max}] \times [l_{min}, l_{max}] | p_d(\mathbf{x}) < p\}, \end{aligned} \tag{6}$$

missä $a_{min}, a_{max}, l_{min}, l_{max}$ ovat pitoajan ja -lämpötilan ala- ja ylärajat, $p_d(\mathbf{x})$ on ennustettu hylkäystodennäköisyys ja p on hylkäystodennäköisyyden yläraja.

5.4 Löydetyn rintaman arviointi

Yksitavoitteisen optimointialgoritmien vertailussa riittää verrata, kuinka lähelle todellista optimia algoritmi pääsee, ja kuinka nopeasti. Koska monitavoiteoptimoinnissa pyritään löytämään useita erilaisia Pareto-optimaalisia vaihtoehtoja, on Pareto-rintamaa kohti suppenemisen lisäksi myös arvioitava saavutetun dominoimattoman rintaman kattamaa alaa ja tasaisuutta. Seuraavaksi esitellään muutama metriikka dominoimattomien rintamien arviointiin. Q, A ja B ovat algoritmin löytämiä dominoimattomia rintamia ja P^* on Pareto-rintama.

5.4.1 Joukkopeittometriika

Joukkopeittometriikka (Set Coverage Metric) $C(A, B)$ mittaa kuinka suuri osa joukon B alkiosta on jonkin joukon A alkion dominoima, eli

$$C(A, B) = \frac{|\{b \in B | \exists a \in A : a \preceq b\}|}{|B|}.$$

Jos kaikki joukon B alkiot ovat joukon A dominoimia niin $C(A, B) = 1$. Vastaavasti $C(A, B) = 0$ kun mikään joukon B alkio ei ole dominoitu. C ei kuitenkaan ole symmetrinen[11], joten on hyvä laskea sekä $C(A, B)$ että $C(B, A)$. Sen etuna on riippumattomuus kohdemuuttujien skaalauksesta, ja se että rintamien arviointiin ei tarvitse tietoa todellisesta Pareto-rintamasta. Toisaalta se antaa tietoa vain joukkojen A ja B keskinäisestä paremmuudesta, ei siitä kuinka lähellä Pareto-rintamaa ne ovat tai kuinka hyvin ratkaisut ovat

[11] $C(A, B) = 1 - C(B, A)$ ei ole välttämättä tosi.

levittyneet. Jos todellinen Pareto-rintama on tiedossa, voidaan yksittäisen algoritmin tehokkuutta arvioida käyttämällä $A = P^*$ ja $B = Q$. [1]

5.4.2 Dominoitu hypertilavuus

Dominoidulla hypertilavuudella mitataan rintaman $Q \in \mathbb{R}^M$ dominoimaa aluetta kohdeavaruudessa. Olkoon $\mathbf{w}$ referenssipiste tavoiteavaruudessa. Tyypillisesti valitaan piste, jonka komponentit koostuvat kunkin kohdefunktion huonoimmista arvoista, eli $w_i = \max_{x \in S} f_i(x)$ kaikilla $i = 1, \ldots, M$. Olkoon

$$V_q = \prod_{i=1}^{M} [q_i, w_i]$$

pisteen $\mathbf{q} \in Q$ ja referenssipisteen välille viritetty M-ulotteinen suorakulmio. Tällöin rintaman Q dominoima hypertilavuus on

$$HV(Q) = \text{tilavuus}(\bigcup_{\mathbf{q} \in Q} V_q).$$

Toisin kuin joukkopeittometriikka, hypertilavuus on riippuvainen kohdearvojen skaalauksesta. Jos kohdefunktioiden arvot ovat eri funktioilla eri kokoluokkaa, myös niiden vaikutus hypertilavuuteen on eri luokkaa. Tämän takia on arvot on hyvä normalisoida ennen hypertilavuuden laskemista. Mikäli todellinen Pareto-rintama on tiedossa voidaan normalisoinnin lisäksi tai sijaan käyttää suhteellista hypertilavuutta

$$HVR(Q) = \frac{HV(Q)}{HV(P^*)}.$$

[1]

5.4.3 Sukupolvietäisyys

Sukupolvietäisyydellä (Generational Distance) mitattaan kuinka kaukana löydetty rintama Q on Pareto-rintamasta, ja määritellään

$$GD(Q) = \frac{1}{|Q|} \sqrt{\sum_{q \in Q} d_q^2},$$

missä $d_q = \min_{p \in P^*} \|q - p\|$. Se on intuitiivisesti selkein mitta suppenemiselle, mutta vaatii että todellinen pareto-rintama tunnetaan. Se ei kuitenkaan

anna tietoa rintaman monimuotoisuudesta, sillä suppeakin rintama voi saada pienen arvon kunhan se on lähellä jotakin osaa Pareto-rintamasta. Sen käänteinen versio

$$IGD(Q) = \frac{1}{|P^*|}\sqrt{\sum_{p\in P^*} D_p^2},$$

missä $D_p = \min_{q\in Q} \|q - p\|$, on tässä suhteessa parempi, koska tällöin arvo riippuu myös siitä kuinka suuren osan Pareto-rintamasta rintama Q kattaa .

5.4.4 Sironta

Sirontamitta (Spread Metric) määritellään

$$S = \frac{\sum_{m=1}^{M} d_m^e + \sum_{q\in Q} |d_q - \bar{d}|}{\sum_{m=1}^{M} d_m^e + |Q|\bar{d}},$$

missä d_q on alkion q etäisyys lähimpään naapuriinsa rintamassa Q, $\bar{d}$ on näiden etäisyyksien keskiarvo ja d_m^e on kussakin tavoitteessa f_m äärimmäisen (maksimaalisen tai minimaalisen) Pareto-rintaman pisteen ja rintaman Q vastaavan pisteen etäisyys. Mikäli rintaman Q havainnot ovat pienessä ryppäässä äärimmäisten termien vaikutus kasvaa suuremmaksi kuin poikkeamat keskimääräisestä etäisyydestä. Tämä tarkoittaa, että sirontamitta mittaa sekä löydetyn rintaman kattavuutta että tasaisuutta.[1].

6 Tulokset

6.1 Hylkäystodennäköisyysmalli

6.1.1 Tutkimusmenetelmät ja aineiston kuvaus

Tässä tutkimuksessa käsitellään teräslevyjä, jotka lämpökäsitellään joko normalisoimalla tai nuorruttamalla. Näiden mallintamisesta on eniten hyötyä asiakkaalle, koska näiden tuotantovolyymi on suuri. Harvinaisemmista lämpökäsittelyistä oli niin vähän dataa, että niiden mallintamista erikseen ei nähty vaivan arvoiseksi, ja niiden käsittely nuorrutettujen ja normalisoitujen levyjen kanssa olisi saattanut vähentää ennustustarkkuutta.

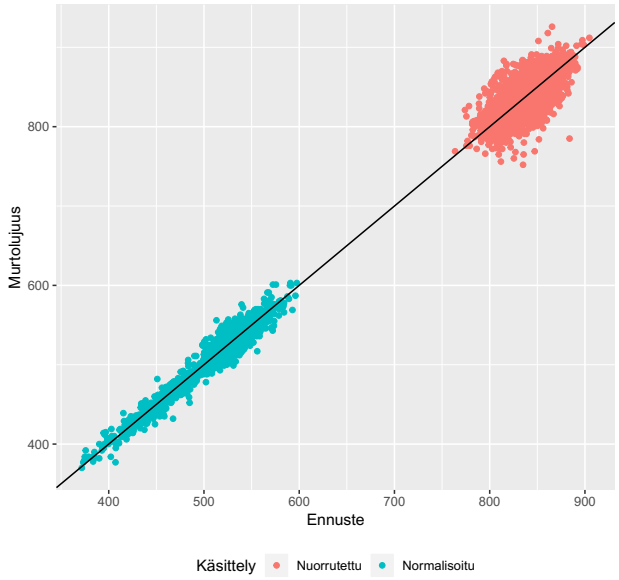
Koska nuorrutetuilla ja normalisoiduilla levyillä on huomattava ero murtolujuuden suhteen, ja koska prosessit ovat melko erilaisia, päätettiin luoda kummallekin ryhmälle oma murtolujuus- ja iskusitkeysmalli. Tuoteryhmien välinen ero voidaan nähdä kuvasta 4. Yhteensä luotiin neljä GBM mallia ennustamaan mittatluosten odotusarvoa ja neljä GAM-mallia ennustamaan GBM-mallien hajontaa.

Vaste	Käsittely	Havaintoja	Muuttujia	RMSE	Korrelaatio
Sitkeys	Nuorrutus	10072	28	36.54	0.64
Sitkeys	Normalisointi	17524	23	22.67	0.93
Lujuus	Nuorrutus	10251	24	18.65	0.64
Lujuus	Normalisointi	19122	22	8.12	0.97

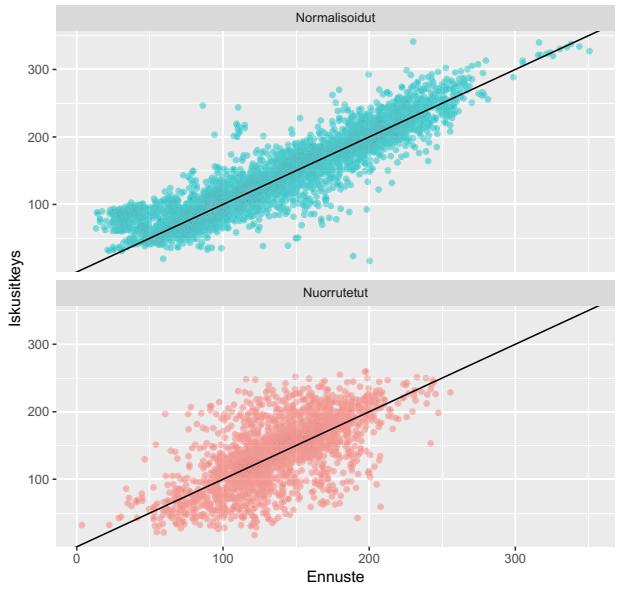
Taulukko 1: GBM mallien ainestojen koot, malleihin valikoituneiden muuttujien määrät, mallin keskineliövirheen juuri (RMSE[12]) ja ennusteen korrelaatio todellisiin arvoihin testiryhmässä.

Taulukossa 1 on esitetty GBM-mallien opetukseen käytettyjen ainestojen rivimäärät. Kukin aineisto jaettiin viiteenkymmeneen yhtä suureen osaan, siten että osat eivät olleet ajallisesti päällekkäin. Näistä osista joka viides valittiin testiryhmään alkaen neljännestä osasta. Tämä tehtiin, jotta opetus ja testiaineisto saataisiin mahdollisimman riippumattomaksi toisistaan. Samasta sulatuksesta tulee useita eri aihioita joiden ominaisuudet eivät ole riippumattomia toisistaan, joten jos saman sulatuksen aihioita on sekä opetus että testiaineistossa saattaa malli ylioppia sulatuksiin liittyvää satunnaisvaihtelua. Koska aineistossa ei ollut aihioiden sulatustietoja saatavilla, oli parempi käyttää aikalohkoihin jakoa täyden satunnaistuksen sijaan. Testiryhmää käytettiin arvioimaan kuinka monta muuttujaa kuhunkin malliin

[12]Root Mean Squared Error



Kuva 4: Murtolujuuden ennusteet ja todelliset arvot testiryhmissä. Nuorrutettujen ja normalisoitujen levyjen lujuusero on hyvin selkeä.



Kuva 5: Iskusitkeyden ennusteet ja todelliset arvot testiryhmissä.

jätettiin. Alunperin aineistossa oli yli 400 muuttujaa, joista lopullisiin malleihin karsiintui yhteensä vajaa 50 eri muuttujaa. Koska murtolujuuteen ja iskusitkeyteen vaikuttavat osittain eri tekijät, itse GBM-malleihin valikoitui 22-28 vaikuttavaa muuttujaa mallista riippuen. Hajontamallit tehtiin paljon yksinkertaisemmiksi ja niissä oli mallista riippuen 7-10 muuttujaa.

6.1.2 Ennustustarkkuus

Kuvissa 4 ja 5 on murtolujuuden ja iskusitkeyden odotusarvoennusteita testiryhmän levyille on verrattu näiden havaittuihin arvoihin. Kuvista nähdään, että normalisoiduilla levyillä sekä murtolujuuden että iskusitkeyden suhteen ennusteet ovat tarkempia. Kuvaajien ryhmittäinen RMSE ja korrelaatio löytyvät taulusta 1. Kuvissa 6 ja 7 optimointiin valitun lämpökäsittelyyn liittyvän lämpötilan ja pitoajan vaikutus ennusteeseen on piirretty mustalla viivalla, ja taustalla on piirrettynä opetusaineiston pisteiden arvo muuttujalle ja vasteen todellinen arvo. Näistä nähdään, että normalisoitujen levyjen malleissa valittujen muuttujien vaikutus ennusteeseen on pientä suhteessa opetusaineiston kokonaisvaihteluun. Etenkin murtolujuuden suhteen vaikutus on lähes olematonta. Nuorrutetuilla muuttujilla vaikutus on selkeästi isompi, vaikkakin voidaan nähdä että näillä kahdella muuttujalla ei voida yksistään selittää suurinta osaa kokonaisvaihtelusta. Kuvaajista nähdään myös optimoinnin syy, eli se, että korkeampi lämpötila ja pitoaika lisäävät sitkeyttä, mutta madaltavat lujuutta.

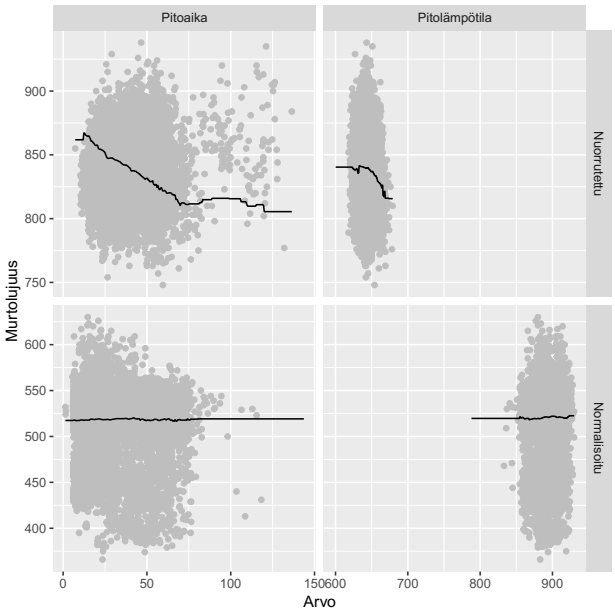
6.1.3 Hajontamallien vertailu

Hylkäystodennäköisyysmallia varten eri hajontamallien ennustuskykyä vertailtiin kullekin mallille. Vertailtavina olivat vakiohajontamalli, jossa hajonta on sama mallin kaikille ennusteille, vastetta $\hat{\varepsilon}_i^2$ ennustava GAM-malli, ja vastetta $\hat{\varepsilon}_i^2(1+\Delta)$ ennustava GAM-malli, joka esiteltiin kappaleen 4.2 lopussa. Kuvassa 8 on arvioitu eri hajontamallien vaikutusta hylkäystodennäköisyyden mallintamiseen käyttäen negatiivisen log-uskottavuuden keskiarvoa[13]

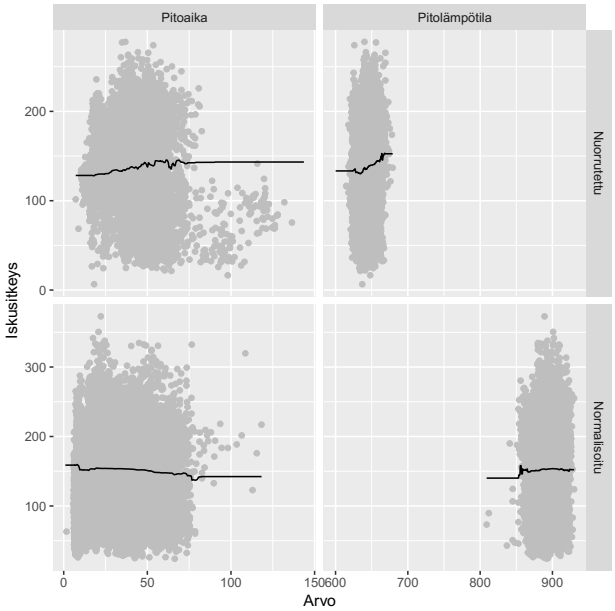
$$AVG(negloglik) = \frac{1}{N_T} \sum_{i \in \mathbf{T}} \left[\frac{1}{2} \log\left(2\pi\hat{\sigma}_i^2\right) + \frac{1}{2} \frac{(y_i - \hat{\mu}_i)^2}{\hat{\sigma}_i^2} \right]$$

missä $\mathbf{T}$ on testijoukkoa vastaavat indeksit, N_T on testijoukon koko, $\hat{\sigma}_i$ on ennustettu hajonta, $\hat{\mu}_i$ ennustettu keskiarvo ja y_i vasteen todellinen arvo havainnolle i. Mitä pienempi ANLL arvo on sitä paremmin malli ennustaa. Kuvassa 8 punainen pylväs on vakiohajontamallin ero nollamalliin, jossa sekä

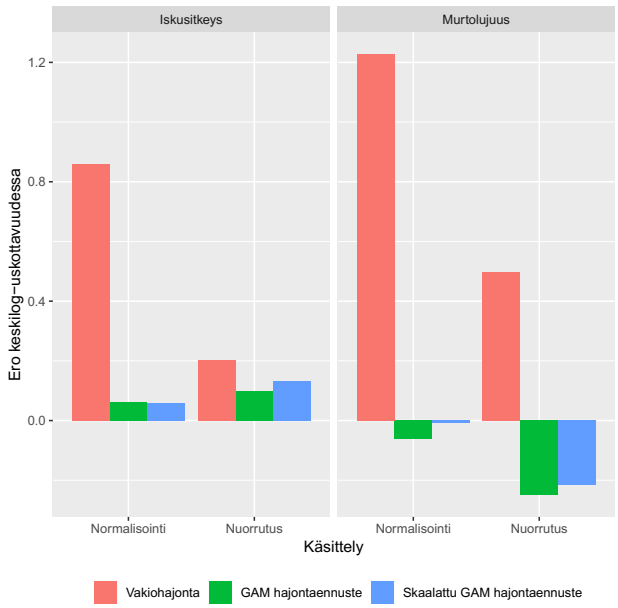
[13] Average Negative Log Likelihood, ANLL



Kuva 6: Lämpökäsittelyn keston ja levyn loppulämpötilan vaikutus murtolujuuden ennusteeseen GBM malleissa. Taustalla harmaalla opetusdatan pisteet.



Kuva 7: Lämpökäsittelyn keston ja levyn loppulämpötilan vaikutus iskusitkeyden ennusteeseen GBM malleissa.



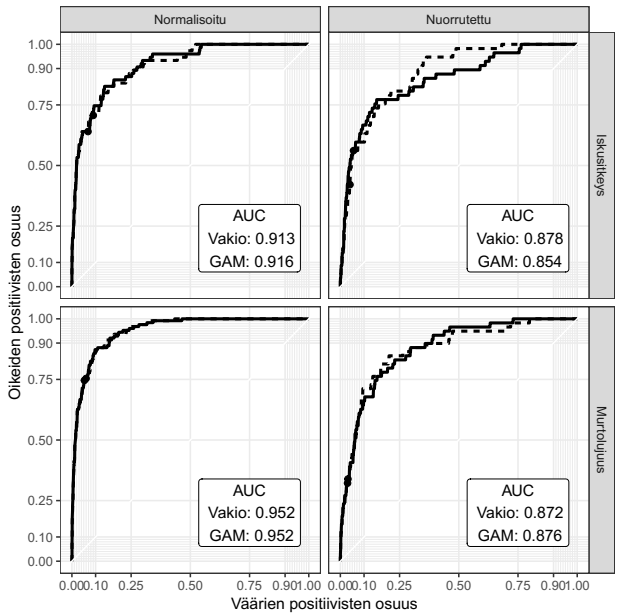
Kuva 8: Eri hajontamallien erot keskilog-uskottavuudessa.

hajonta että odotusarvo oletetaan vakioiksi. Muut ovat eroja vakiohajontamalliin. Positiivinen arvo kuvaajassa tarkoittaa että malli on parempi kuin se malli johon verrataan. Kuvasta nähdään että etenkin nuorrutettujen iskusitkeysmalli hyötyy GAM-hajontamllista, mutta murtolujuuden mallinnuksessa vakiohajontamalli antaa parhaan tuloksen. Hajontaestimaatin skaalaus arvolla $1+\Delta$ paransi uskottavuutta skaalaamattomaan GAM ennusteeseen nähden kaikissa paitsi normalisoidun materiaalin iskusitkeysmallissa.

6.1.4 ROC-käyrät

Mallien ennustuskykyä on lisäksi kuvassa 9 arvioitu ROC-käyrien[14] avulla. ROC-käyrä kuvaa sitä kuinka hyvin malli erottelee hylkäykset ja hyväksytyt. Mitä lähempänä käyrä on vasenta ylänurkkaa sitä paremmin malli osaa erotella. Jos käyrä on lähellä kuvaajan halkaisijaa, mallin erottelukyky on huono tai jopa olematon. AUC-arvo[15] on käyrän alapuolella olevan alueen pinta-ala, joka kertoo erottelukyvystä tiivistettynä. Kuvaajien perusteella kaikki mallit erottelevat hylkäykset suhteellisen hyvin, mutta normalisoitujen levyjen mallit erottelavt hylkäykset nuorrutettujen levyjen malleja paremmin. On huomattava kuitenkin, että kuvaajan muotoon valittu hylkäysraja.

[14]Receiver operating characteristic curve
[15]Area under curve



Kuva 9: Hylkäysennustemallien ROC-käyrät. Katkoviiva kuvaa vakiohajontamallia ja musta viiva GAM hajontamallia.

6.2 Evoluutioalgoritmien vertailu

6.2.1 Tutkimusmentelmän kuvaus

Monitavoiteoptimointialgoritmeja vertailtiin suorittamalla kummallekin algoritmille 100 ajoa 50 sukupolveen asti, jossa etsittiin Pareto-optimaalisia pisteitä eräälle aineistosta valitulle nuorrutetulle levylle, kun hylkäystodennäköisyys on pienempi kuin 0.05. Haettu pitolämpötila rajattiin alueelle 620-675 ja haettu pitoaika alueelle 12-130 minuuttia, jotta pysyttäisiin opetusaineiston alueella. Kunkin ajon joka sukupolven dominoimattomalle rintamalle laskettiin eri rintamamittoja, ja ajojen keskiarvot joka sukupolvelle on piirretty kuvaajiin 13, 14 ja 15. Kuvaaja 13 ja 14 sisältävät samat tiedot sukupolvittain ja algoritmin ajoajan suhteen. Kuvaajassa 15 on ainoastaan algoritmien keskinäinen joukkopeittometriikka. Taulukossa 2 on esitelty ajojen viimeisien sukupolvien keskiarvot.

DEMO-algoritmista käytettiin 20 alkion ryhmille rinnakkaistettua versiota, koska käytetyn GBM-mallin ennusteen laskeminen on tehokkaampaa usealle pisteelle kerralla kuin kaikille erikseen. Näin pyrittiin vähentämään rinnakkaistettavuuden vaikutusta suoritusnopeuteen.

Näiden algoritmien lisäksi käytettiin vertailukohdaksi brute force menetelmää, jossa hylkäystodennäköisyys arvioitiin kaikille optimoitavien para-

metrien jakokohdille GBM mallissa, ja näistä pisteistä haettiin kelvollisien pisteiden Pareto-optimaalinen rintama. Tämän rintaman avulla laskettiin suhteellinen hypertilavuus ja joukkopeittometriikka kummankin algoritmin sukupolvien suhteen. Kumpikaan vertailtava algoritmi ei saavuttanut yhtään vertailukohdan pistettä, minkä johdosta joukkopeittometriikka sen suhteen on molemmilla 1 kaikissa sukupolvissa. Tämän takia vertailurintaman joukkopeittometriikoita ei ole esitetty kuvaajissa. Brute force menetelmän ajoaika oli noin 40 sekuntia.

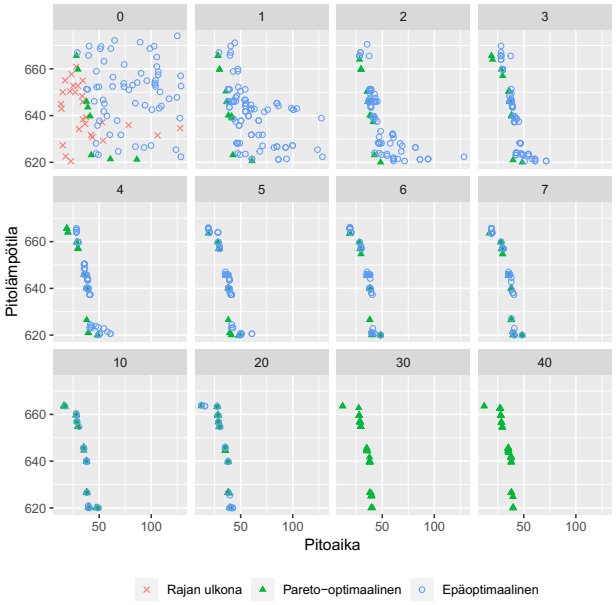
Ajoja suoritettiin myös satunnaisesti valituille levyille, mutta usein hylköystodennäköisyys oli joko niin korkea tai niin matala, että joko koko hakualue oli kelvollinen, jolloin optimointi on triviaali (valitaan pienin arvo kummastakin parametrista), tai kelvollisia pisteitä ei ollut ollenkaan, jolloin algoritmit supistuvat kohti pienintä hylkäystodennäköisyyttä eikä varsinaista rintamaa synny. Tässä työssä demonstroidaan MO-algoritmien käyttöä vain yhdelle levylle, koska MO-algoritmien käyttäminen tällaisille levyille ei ole välttämättä järkevää.

Algoritmi	Aika	C	GD	HVR	IGD	spread
DEMO	11.94	0.38	0.0049	0.9977	0.0040	0.0061
NSGA-II	7.52	0.21	0.0021	0.9959	0.0094	0.0329
Vertailu	40.57	1.00	0.0000	1.0000	0.0000	0.0000

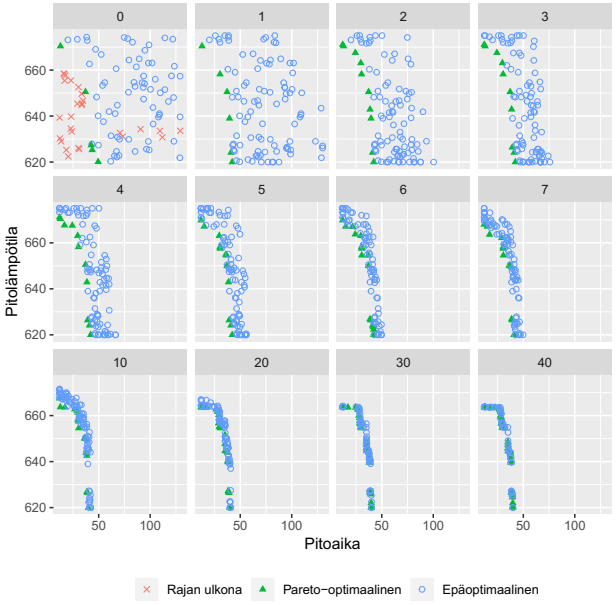
Taulukko 2: Viimeisien sukupolvien tunnusluvut. Ajoaikaa lukuun ottamatta mitä lähemmäksi vertailuarvoja päästään sitä parempi.

6.2.2 Algoritmien tuloksien erot

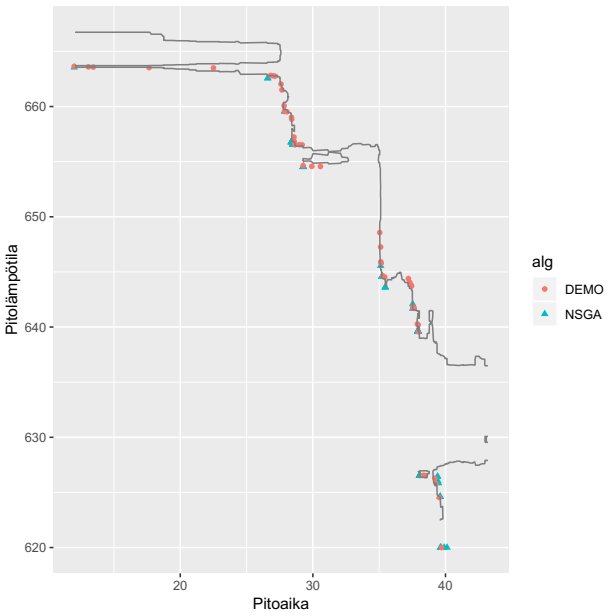
Kuvissa 10 ja 11 on esitelty kummallekin algoritmille yhden testiajon sukupolvien populaatioita. Kuvasta 10 voidaan havaita, että NSGA karsii huonoja ratkaisuja hyvin nopeasti ja päätyy kapeaan mutta epätasaiseen rintamaan. Vastaavasti kuvasta 11 nähdään että DEMO säilyttää ratkaisuja pidempään, mutta sen lopputulos on tasaisempi. Kuvaaja 12 kuvaa näiden ajojen viimeisien sukupolvien dominoimattomat rintamat brute force-ajon hylkäystodennäköisyyden 0.05 tasa-arvokäyrää vastaan. Kumpikin algoritmi pääsee hyvin lähelle todellista rintamaa. Kuvasta nähdään myös että tasa-arvokäyrän epätasaisuuden takia Pareto-rintama on erittäin katkonainen, minkä johdoista ratkaisuja ei joiltakin alueilta löydy. NSGA:n ja DEMO:n tulokset ovat niin lähellä toisiaan että suurestakin kuvasta on vaikea arvioida silmällä kumpi on päässyt lähemmäksi todellista rintamaa. Sukupolvien rintamametriikoista (kuva 13) voidaan kuitenkin nähdä että DEMO:n lopputulos on keskimää-



Kuva 10: Sukupolvien kehitys NSGA-II algoritmilla.

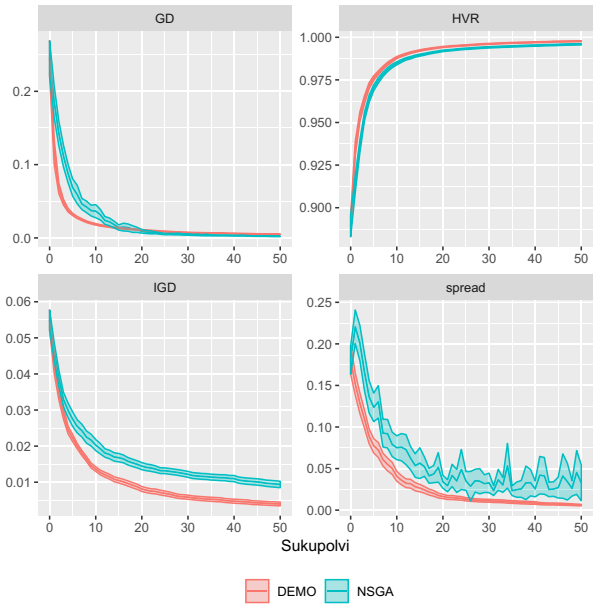


Kuva 11: Sukupolvien kehitys DEMO algoritmilla

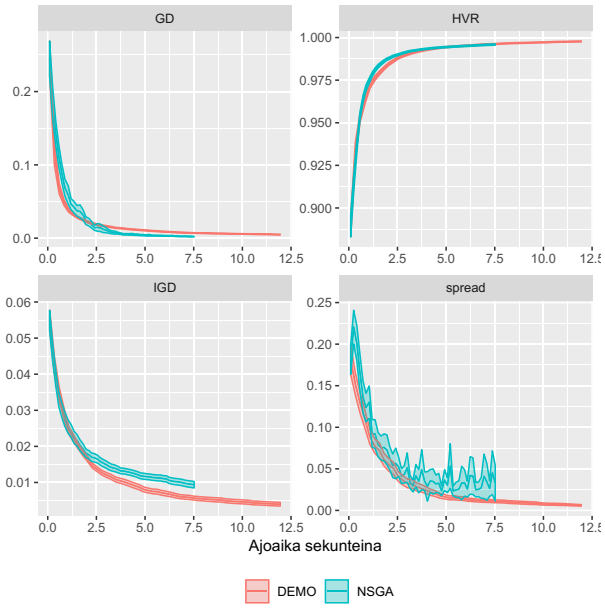


Kuva 12: Kummankin MO-algoritmin viimeisen sukupolven dominoimattoman rintaman pisteet. Käyrä on rajoitefunktion 0-käyrä, joka vastaa hylkäustodennäköisyyttä 0,05.

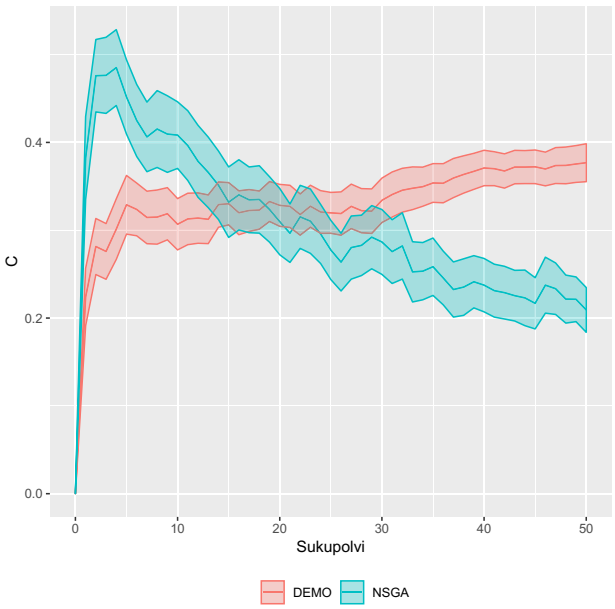
rin NSGA:ta parempi kaikissa metriikoissa paitsi tavanomaisessa sukupolvietäisyydessä (GD). Tästä kuvasta huomataan myös että DEMO:n tuloksissa on vähemmän satunnaisvaihtelua kuin NSGA:n tuloksissa, etenkin sirontamitan (spread) suhteen. Kuvassa 14 samat luvut on kuvattu sen mukaan kuinka kauan kyseisen sukupolven saavuttamiseen kultakin algoritmilta meni keskimäärin. Suoritusaikaan suhteutettuna tilanne on hieman tasiasempi. NSGA:lla on etu hypertilavuudessa (HVR) aikaisemmilla sukupolvilla, mutta DEMO ottaa sen kiinni loppua kohden. Kuvasta 15 nähdään, että NSGA:n sukupolvien ratkaisut dominoivat suurta osaa DEMOn ratkaisuista ensimmäisillä sukupolvilla, mutta ajon jatkuessa dominoitujen ratkaisujen osuus vähenee ja DEMO:n löytämän rintaman dominoima osuus NSGA:n rintamasta kasvaa.



Kuva 13: Rintamamittojen kehitys sukupolvittain. Pienempi arvo on parempi, paitsi hypertilavuudella (HVR).



Kuva 14: Rintamamittojen kehitys algoritmin suoritusajan mukaan. Pienempi arvo on parempi, paitsi hypertilavuudella (HVR).



Kuva 15: Mallien sukupolvien keskinäiset joukkopeittometriikat. Suurempi arvo on parempi.

7 Pohdinta

7.1 Käytännön rajoitukset

Teolliset massatuotantoprosessit pyritään yleensä ajamaan mahdollisimman samalla tavalla joka kerta kun käsitellään samanlaista tuotetta. Tämän johdosta on tyypillistä, että tutkittava data on kasautunut ryppäisiin, joiden ulkopuolelta on erittäin vaikea tehdä ennusteita. Etenkin harvinaisten ja selkeästi poikkeuksellisten tuotteiden kohdalla ennustaminen on hankalaa. Tämän lisäksi laatupoikkeamien ennustamisessa on ongelmana se että niitä on tyypillisesti vähän, mikä tarkoittaa että vaikka hyvästä datasta olisi esimerkkejä riittävästi, ei ennustettavasta ongelmasta ole välttämättä riittävästi tietoa.

Monitavoiteoptimoinnin osalta normalisoidut levyt jätettiin enimmäkseen huomiotta. Syyt tähän juontuvat normalisointiprosessin ominaisuuksista. GBM-malleissa optimoitavien parametrien vaikutus ennusteeseen oli niin pientä, että optimoinnin vaikutus ei olisi edes havaittavissa (kuvat 6 ja 7). Tämä on tyypillistä normalisoinnille, eikä sinällään puute malleissa. Pienestä vaikutuksesta johtuen suurimmalle osalle levyistä hylkäystodennäköisyys oli joko liian pieni, tai liian suuri että kelvollisten ratkaisujen joukko sisältäisi pisteitä, sekä eroaisi hakuavaruudesta valitulla hylkäystodennäköisyysrajalla.

Niille normalisoiduille levyille, joilla kelvollisien ratkaisujen joukko oli epätriviaali, pareto-optimaalinen rintama oli lähes lämpötila-akselin suuntainen. Tämä johtuu siitä, että pitoajan vaikutus lopputulokseen oli lähes olematonta normalisoiduilla levyillä. Näiden syiden lisäksi normalisoiduissa tuotteissa ei iskusitkeyden ja murtolujuuden vastakkaisuus tule kovin hyvin esille, toisin kuin nuorrutetuilla levyillä. Isolla osalla nuorrutettujakaan levyjä ei välttämättä synny monitavoiteoptimoinnin kannalta järkevää kelvollisten ratkaisujen joukkoa, tämä asia korjaantuu hylkäystodennäköisyysrajan muutoksella huomattavasti helpommin kuin normalisoitujen tapauksessa. Tuotteet, joille tämä on tarpeen, ovat usein kuitenkin helpommin valmistettavia teräksiä, joten optimoinnille on vähemmän tarvetta niiden tapauksissa.

7.2 Huomioita NSGA-II:n ja DEMO:n eroista

DEMO:n ja NSGA-II ero on siinä kuinka nopeasti Pareto-rintamasta kauempana olevat pisteet katoavat populaatiosta. Tähän luultavasti pääsyynä on jälkeläisten luonnin erot. Turnausvalinnan puute DEMO:ssa tarkoittaa että jälkeläisiä luodaan kaikille alkioille tasaisesti, ja näin ollen uusia ratkaisuja luodaan laajemmalle alueelle kuin NSGA:ssa. Tämä myös tarkoittaa, että paremmat ratkaisut eivät ole etuasemassa jälkeläisten luonnissa. Jälkeläiskandidaatin ja vanhemman vertailu ja niistä huonomman poisto ennen populaatioon lisäämistä toimii tietyssä mielessä pseudo-turnauksena. Erityissesti algoritmien ero näkyy sirontamitan kehityksessä sukupolvittain. NSGA:lla kehitys on hyvin epätasaista, mikä viittaa siihen, että algoritmi ei aina löydä riittävän kattavaa osaa Pareto-rintamasta, johon todennäköinen syy on liian nopea karsinta. Pareto-rintaman lähelle päästään nopeasti, mutta sen jälkeen populaation monimuotoisuus ei riitä löytämään Pareto-rintaman puuttuvia pisteitä. Evoluutioparametrien, kuten mutatiotodennäköisyyden, arvoja säätämällä voisi olla mahdollista parantaa sironnan kehittymistä.

Yksi osatekijä algoritmien väliseen eroon hypertilavuuden kehityksessä voi olla rajoitefunktion paloittainen muoto. Sen johdosta kelvollinen alue on jokseenkin kulmikas, minkä johdosta Pareto-rintaman pisteet ovat hyvin erillisiä. Tämä ei ole kovin tyypillistä monitavoiteongelmissa. Tyypillisissä monitavoiteoptimoinnin benchmark-ongelmissa etsittävät rintamat ovat suhteellisen sileitä, vaikkakin joskus epäjatkuvia (esim. ZDT3 [14]). Toinen ero tyypilliseen MO-ongelmaan on se, että ongelma on rajoitteen hallitsema, ja kohdefunktiona on pelkkä identiteetti. Koska simuloitu binääriristeymä valitsee jälkeläispisteet vanhempien väliltä, vanhempien ollessa lähellä eri pisteitä Pareto-rintamassa, suurin osa niiden välisistä pisteistä tulee olemaan epäkelpoja rajoitteen sahalaitaisen muodon takia. Tämä yhdistettynä korkeaan elitismiin tarkoittaa, että uusia ratkaisuja ei jää sukupolveen kauhean monta

sen jälkeen, kun on päästy riittävän lähelle rintamaa. Mahdollinen ratkaisu tähän olisi varata joka sukupolveen tilaa huonommille ratkaisuille. Tutkimuksessa käytettyjen päätöspuiden sijaan GBM:n heikkoina oppijoina voitaisiin myös käyttää sileämpiä funktioita, kuten esimerkiksi MARS[16]-splinejä [6]. Tällöin MO-algoritmien ei tarvitsisi etsiä erillisiä pisteitä, vaan yhtenäisiä rintamia.

NSGA-II:n ja demon suoritusajoissa oli huomattava ero, mutta on mahdollista että NSGA-II:lle käytetty funktio oli paremmin optimoitu kuin DEMO:lle käytetty funktio. On myös otettava huomioon että DEMO saattaa suoriutua NSGA:ta paremmin suuremmilla populaatioilla, sillä se käyttää samaa algoritmia ($\mathcal{O}(n^2)$ kompleksisuus) kuin NSGA ja karsittava rintama on usein pienempi kuin NSGA:lla. Sillä kuinka monta ulottuvuutta optimointiongelmalla on ei pitäisi olla suurta vaikutusta optimointimenetelmien keskinäiseen eroon suoritusnopeudessa.

7.3 Mitä jäi testaamatta

Tässä tutkimuksessa aineisto jaettiin kahtia nuorrutettuihin ja normalisoituihin levyihin koska niiden käytös oli hyvin erilaista murto- ja myötölujuuden suhteen, mutta iskusitkeyden suhteen ei näillä levyillä ei ole kovin suurta eroa. Ei ole tutkittu voiko iskusitkeyttä ennustaa yhdellä mallilla paremmin kuin kahdella, mutta prosessit eroavat niin paljon, etenkin tuotteen lopputlämpötilan suhteen, että tämä ei ole todennäköistä.

MO-algoritmejen laajempi testaus suoritettin vain yhdenlaisella ongelmalla. Esimerkiksi moniulotteisempaa optimointia ei testattu kuin pintapuolisesti, ja eri evoluutioparametrien vaikutuksia ei vertailtu. Suorituskyvyn vertailun kannalta tämä olisi ollut hyödyllistä. Tämän lisäksi optimointiajot suoritettiin vain 50. sukupolveen asti, eikä testattu kuinka monta sukupolvea algoritmeilta kestää löytää Pareto-rintama.

Tutkimuksesta jäi puuttumaan käytännön kokeilu, eikä näin ollen ole varmaa miten mallin soveltaminen todelliseen tuotantoon onnistuisi. Tämän lisäksi, koska tutkimusongelman ratkaisuun käytettiin menetelmiä usealta eri alalta, ei kaikkia asioita välttämättä pystytty esittelemään kovin syvällisesti.

[16]Multivariate Adaptive Regression Spline

8 Yhteenveto

Tutkielmassa esiteltiin monitavoiteoptimointi, evoluutioalgoritmit ja näiden yhdistäminen monitavoitteisiksi evoluutioalgoritmeiksi, sekä datapohjainen hylkäystodennäköisyyden ennustamisen menetelmä. Näitä käytetttiin teräksen lämpökäsittelyn pareto-optimaalisten asetusten löytämiseen, siten että hylkäystodennäköisyyden mallin ennusteita käytettiin monitavoiteoptimoinnin rajoitefunktiona. Monitavoitteisia evoluutioalgoritmeja NSGA-II ja DEMO vertailtiin tämän rajoitetun optimointiongelman ratkaisemisessa nopeuden ja löydetyjen ratkaisujoukkojen monimuotoisuuden suhteen. Lisäksi vertailukohtana käytettiin brute force algoritmilla haettua pareto-optimaalista joukkoa.

Kaiken kaikkiaan havaittiin että monitavoiteoptimointi on sovellettavissa nuorrutusprosessin optimointiin, mutta käytetty menetelmä vaatii parannuksia, että siitä saataisiin mahdollisimman hyödyllinen. Hylkäystodennäköisyyden mallinnus toimi suhteellsen hyvin optimointialgoritmien kanssa, tosin keskiarvoennusteen mallin epäsileys teki haettavasta Pareto-rintamasta hankalan löytää. DEMO osoittautui tässä tutkimusongelmassa tarkemmaksi algoritmiksi, mutta NSGA-II:n nopeus on sille eduksi, ja sileämmällä rajoitefunktiolla se saattaisi suoriutua paremmin.

Hylkäystodennäköisyyden ennustamisessa onnistuttiin myös hyvin ja hylkäysriskirajaa säätämällä asiakas voi valita kuinka varovainen hän haluaa prosessin arvojen säädön kanssa olla.

Lähdeluettelo

[1] Kalyanmoy Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: John Wiley & Sons, Inc., 2001. ISBN: 047187339X.

[2] Kalyanmoy Deb et al. "A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II". Teoksessa: *Parallel Problem Solving from Nature PPSN VI*. Toim. Marc Schoenauer et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, s. 849–858. ISBN: 978-3-540-45356-7.

[3] James S. Dyer et al. "Multiple Criteria Decision Making, Multiattribute Utility Theory: The Next Ten Years". *Management Science* 38.5 (1992), s. 645–654. DOI: 10.1287/mnsc.38.5.645. eprint: https://doi.org/10.1287/mnsc.38.5.645. URL: https://doi.org/10.1287/mnsc.38.5.645.

[4] A. E. Eiben ja J.E. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag Berlin Heidelberg, 2003.

[5] Jerome H. Friedman. "Greedy Function Approximation: A Gradient Boosting Machine". *The Annals of Statistics* 29.5 (2001), s. 1189–1232. ISSN: 00905364. URL: http://www.jstor.org/stable/2699986.

[6] Jerome H. Friedman. "Multivariate adaptive regression splines". *Ann. Statist* (1991).

[7] Jerome H. Friedman. "Stochastic Gradient Boosting". *Comput. Stat. Data Anal.* 38.4 (helmikuu 2002), s. 367–378. ISSN: 0167-9473. DOI: 10.1016/S0167-9473(01)00065-2. URL: http://dx.doi.org/10.1016/S0167-9473(01)00065-2.

[8] J. A. Harding et al. "Data Mining in Manufacturing: A Review". *J. Manuf. Sci. Eng* 128.4 (joulukuu 2005), s. 969–976. ISSN: 1087-1357. DOI: 10.1115/1.2194554. URL: http://dx.doi.org/10.1115/1.2194554.

[9] Trevor Hastie ja Robert Tibshirani. "Generalized Additive Models". *Statistical Science* 1.3 (1986), s. 297–310. ISSN: 08834237. URL: http://www.jstor.org/stable/2245459.

[10] Ilmari Juutilainen, Satu Tamminen ja Juha Röning. "A Tutorial to Developing Statistical Models for Predicting Disqualification Probability". Teoksessa: *Computational Methods for Optimizing Manufacturing Technology: Models and Techniques*. Toim. J. Paulo Davim. IGI Global, 2012, s. 368–399. DOI: 10.4018/978-1-4666-0128-4.ch015.

[11] Lindroos, Sulonen ja Veistinen. *Uudistettu Miekk-ojan metallioppi*. Otava, 1986.

[12] Kaisa M. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, 1999.

[13] Andres Munoz. "Machine learning and optimization" (2014). URL: https://pdfs.semanticscholar.org/7fbb/a79630b5a09dd66ab13f00c3aefaa56cf268.pdf.

[14] Tea Robič ja Bogdan Filipič. "DEMO: Differential Evolution for Multiobjective Optimization". Teoksessa: *Evolutionary Multi-Criterion Optimization*. Toim. Carlos A. Coello Coello, Arturo Hernández Aguirre ja Eckart Zitzler. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, s. 520–533. ISBN: 978-3-540-31880-4.

[15] Stuart J. Russell ja Peter Norvig. *Artificial Intelligence: A Modern Approach*. International 2nd. Pearson Education, 2003. ISBN: 0130803022.

[16] *Teräskirja*. Metallinjalostajat ry, 2014.

[17] Jyrki Wallenius et al. "Multiple Criteria Decision Making, Multiattribute Utility Theory: Recent Accomplishments and What Lies Ahead". *Management Science* 54.7 (2008), s. 1336–1349. DOI: 10.1287/mnsc.1070.0838. eprint: https://pubsonline.informs.org/doi/pdf/10.1287/mnsc.1070.0838. URL: https://pubsonline.informs.org/doi/abs/10.1287/mnsc.1070.0838.

[18] Simon N. Wood. *Generalized Additive Models: an introduction with R*. Chapman & Hall/CRC Press, 2006.